UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería Informática

Minería de datos aplicada a la detección de displasias corticales

Data mining applied to the detection of cortical dysplasias

Realizado por
Juan Francisco Gutiérrez Manzanares

Tutorizado por
Rafael Morales Bueno
José del Campo Ávila

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA INFORMÁTICA

Minería de datos aplicada a la detección de displasias corticales

Data mining applied to the detection of cortical dysplasias

Realizado por
Juan Francisco Gutiérrez Manzanares

Tutorizado por
Rafael Morales Bueno
José del Campo Ávila

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2021

Fecha defensa: julio de 2021

Resumen

La displasia cortical focal (DCF) es la principal causa de epilepsia tanto en niños como en adultos. La resonancia magnética es una herramienta ideal para detectar el DCF ya que destaca por sus propiedades como la no invasividad y la capacidad de producir imágenes de alta resolución. Las lesiones de la DCF varían en tamaño, forma y ubicación para diferentes pacientes y hacen que la detección manual requiera mucho tiempo y deba ser realizada por un experto. La segmentación automática de las lesiones DCF es un desafío debido a la diferencia en la intensidad de la señal en las imágenes adquiridas con diferentes máquinas, ruido y otros tipos de distorsiones. La mayoría de los métodos propuestos en la literatura utilizan técnicas convencionales de procesamiento de imágenes y aprendizaje automático en las que su precisión se basa en las características extraídas de las imágenes con las que se entrena. Por lo tanto, la extracción de características debe realizarse con la mayor precisión posible, lo que requerirá la inclusión de conocimiento por parte del experto. La capacidad de aprender las características y representaciones que queremos a partir de los datos de entrenamiento sin ningún tipo de intervención humana hace que la red neuronal convolucional (CNN) sea un método adecuado para abordar estos inconvenientes. Por lo que sabemos, este trabajo es de los primeros en utilizar un modelo basado en CNN para resolver el problema mencionado anteriormente utilizando imágenes Flair y T1. En este proyecto se personaliza la popular arquitectura EfficientNet modificando las últimas capas de la red y entrenando el modelo propuesto desde cero (utilizando imágenes de resonancias magnéticas adquiridas con escáneres 1.5T y 3T). La tasa de detección de DCF del modelo propuesto tiene un porcentaje de precisión del 84.466 % sobre el último paciente facilitado.

Palabras clave: Displasia, T1, Flair, Red Neuronal Convolucional, Detector.

Abstract

Focal cortical dysplasia (FCD) is the leading cause of epilepsy in both children and adults. Magnetic resonance imaging is an ideal tool to detect FCD as it stands out for its properties such as non-invasiveness and the ability to produce high resolution images. FCD lesions vary in size, shape and location for different patients and make manual detection time consuming and need to be performed by an expert. Automatic segmentation of FCD lesions is challenging due to the difference in signal intensity in images acquired with different machines, noise, and other types of distortions. Most of the methods proposed in the literature use conventional image processing and machine learning techniques in which their precision is based on the features extracted from the images on which they are trained. Therefore, feature extraction should be done as accurately as possible, which will require the inclusion of knowledge by the expert. The ability to learn the features and representations we want from the training data without any human intervention makes the convolutional neural network (CNN) a suitable method to address these drawbacks. As far as we know, this work is among the first to use a CNN-based model to solve the aforementioned problem using Flair and T1 images. In this project, the popular EfficientNet architecture is customized by modifying the last layers of the network and training the proposed model from scratch (using magnetic resonance images acquired with 1.5T and 3T scanners). The FCD detection rate of the proposed model has a precision percentage of $84,466\,\%$ over the last patient provided.

Keywords: Dysplasia, T1, Flair, Convolutional neural network, Detector.

Índice

1

Introducción

La intención de este capítulo es hacer una introducción de los diferentes aspectos con los que vamos a trabajar a lo largo de este proyecto donde se propone la creación de un sistema que ayude a la detección automática de displasias. En sucesivos capítulos se profundizará en cada uno de ellos.

Nos centraremos en qué son las displasias y su relación con la epilepsia, explicaremos qué es el Data Mining y las Redes Neuronales Convolucionales necesarias para la realización de este trabajo, expondremos el estado de la técnica y la forma en que se obtienen las imágenes de los pacientes. Finalmente concluiremos este capítulo con los objetivos de este trabajo.

1.1. Introducción

La epilepsia es una alteración neurológica crónica, caracterizada por crisis convulsivas recurrentes y espontáneas, producidas por descargas eléctricas anormales de las neuronas corticales. Las descargas se producen debido a un aumento de la excitabilidad, o una depresión de los mecanismos inhibidores de la neurotransmisión.

Desde el punto de vista clínico, las epilepsias se clasifican como parciales, cuando las convulsiones se originan desde un área localizada del cerebro o generalizadas, cuando las convulsiones se originan simultáneamente de ambos hemisferios cerebrales.

Una posible causa de epilepsia es la displasia cortical (que se puede identificar en imágenes radiológicas).

Las displasias corticales corresponden a un tipo especial de alteraciones del desarrollo de la corteza cerebral que constituyen una causa importante de frecuentes trastornos neurológicos y psicológicos. Son un reto importante para el radiólogo, debido a que muchas veces las alteraciones detectadas en las imágenes médicas son sutiles o, incluso, pueden resultar completamente normales. Un ejemplo de resonancias en las que se encuentra este tipo de anomalías

es la Figura 1.

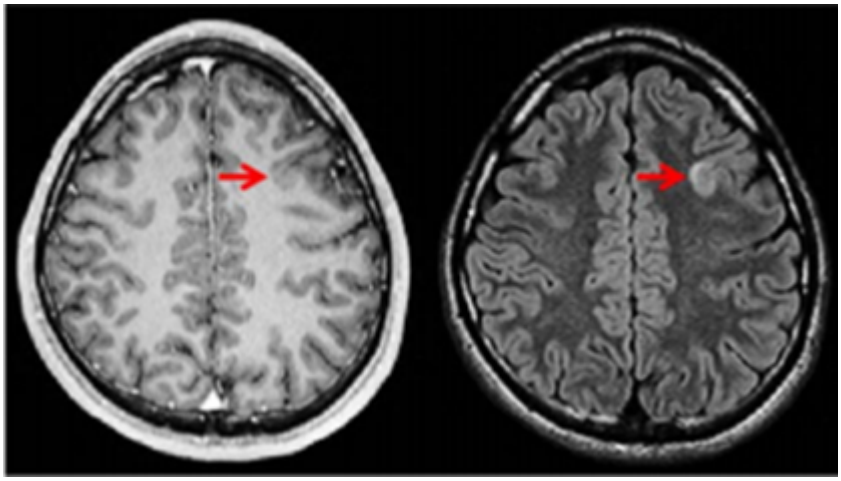

Figura 1: Imágenes de resonancia magnética axial potenciadas en T1 (izquierda) y T2 (derecha) de muestra tomadas de un paciente de 21 años con epilepsia. La displasia cortical focal (flechas rojas) se presenta como una pérdida de contraste blanco grisáceo en las imágenes T1 y como una hiperintensidad en las imágenes T2. [9]

El enorme desarrollo que está viviendo la tecnología asociada a la Inteligencia Artificial (IA) está dando lugar en los últimos tiempos a nuevas herramientas y aplicaciones que abarcan numerosos campos de la ciencia.

Una de las áreas donde los avances han sido más notables es el reconocimiento de imágenes, en parte gracias al desarrollo de nuevas técnicas de Aprendizaje Profundo (Deep Learning). Hoy en día se dispone de sistemas muy precisos que ayudan a los humanos en las tareas de clasificación y detección de lesiones en imágenes, como la detección de lesiones en la piel o la detección de lesiones en mamografías.

Aplicar la Inteligencia Artificial al ámbito médico para el diagnóstico de la epilepsia en la detección de displasias por medio de imágenes parece viable. Es por ello que este proyecto trata de ayudar a estos especialistas a través de técnicas para el procesamiento automático de imágenes.

1.2. Minería de datos

La Minería de Datos (*Data Mining*) es un campo de la estadística y las ciencias de la computación que intenta ayudar a comprender el contenido de los conjuntos de datos.

El data mining trabaja buscando patrones, comportamientos, agrupaciones, secuencias, tendencias o asociaciones que puedan generar algún modelo que nos permita comprender mejor los datos para ayudar en una posible toma de decisión.

Una parte fundamental para realizar cualquier proyecto en el que se use minería de datos es la metodología. Se entiende por metodología de un proyecto al proceso que seguiremos para gestionar nuestras actividades siguiendo unos requisitos y pasos, con el fin de encontrar rutas de trabajo optimizadas.

Para escoger correctamente la metodología de trabajo es necesario tener en cuenta dos aspectos fundamentales que son la gestión de los recursos y el tiempo que se tiene para la realización del proyecto. Además, existen numerosas aplicaciones software que ayudan a la gestión de proyectos para empresas.

Existen diversos tipos de metodologías bastantes populares como pueden ser:

- SEMMA
- CRISP-DM
- ASUM-DM una evolución de CRISP-DM de IBM
- TDSP de Microsoft
- MDP (The Model Development Process)

Hablaremos más delante de cuál va a ser la empleada para este proyecto y el por qué.

1.3. Redes Neuronales Convolucionales

La Red Neuronal Convolucional o CNN es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver".

Las redes convolucionales contienen varias capas ocultas, donde las primeras pueden detectar líneas, curvas y así se van especializando hasta poder reconocer formas complejas como un rostro, siluetas, etc. Se puede observar en la Figura 2 el proceso que debe realizar la CNN para llegar a reconocer formas.

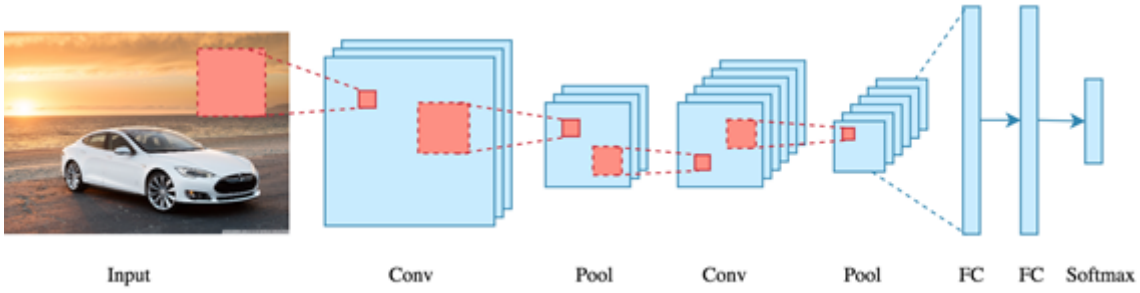

Figura 2: Proceso de una red Neuronal Convolucional. [8]

Las redes neuronales convolucionales (CNN) se emplean para la clasificación de imágenes desde el año 2012.

Uno de los conjuntos de datos más usados para entrenar estas redes neuronales convolucionales es ImageNet, que cuenta con 14.197.122 imágenes repartidas entre 21841 clases diferentes.

La existencia de estas bases de datos junto con los diferentes modelos permite la aparición de casi todo tipo de clasificadores y detectores de imágenes, los más comunes suelen ser los de animales, de personas o de modelos de coches, pero los hay incluso de células o de baches en carreteras.

Sin embargo, para este proyecto no se usarán los datos que facilita ImageNet, ya que las imágenes que contengan displasias cerebrales no son algo común y que se encuentre dentro de este conjunto de datos, así que conseguiremos realizar el entrenamiento de nuestra CNN por medio de las imágenes que nos proporcionen desde el Hospital Regional Universitario de Málaga.

1.4. Obtención de imágenes

Como veníamos recalcando previamente, nuestro modelo no ha sido entrenado con imágenes de un banco de datos ya existente, sino que ha sido entrenado con el conjunto de imágenes facilitado por el Hospital Regional Universitario de Málaga.

Para poder utilizar las imágenes de los pacientes ha sido necesario realizar un Protocolo Ético que necesitó ser aprobado por el Comité de Ética Provincial de Málaga por medio del Portal de Ética de la Investigación Biomédica de Andalucía (PEIBA).

Dicho protocolo consta de algunos documentos entre los que cabe destacar el documento

de confidencialidad, en el cual nos comprometemos tanto mis tutores, Rafael Morales Bueno y José del Campo Ávila, como yo, Juan Francisco Gutiérrez Manzanares, a mantener la confidencialidad de todos los datos de carácter personal de los pacientes de los que utilizamos sus resonancias magnéticas para la realización de este proyecto.

Este trabajo, además, se realizará siguiendo las directrices de las Normas de Buena Práctica Clínica y de la Declaración de Helsinki (Fortaleza 2013) y para preservar la confidencialidad de la información recogida. Se seguirán los criterios para el tratamiento de datos en la investigación en salud según figura en la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. Para ello, hemos garantizado total anonimicidad de las imágenes asegurando así, de cara al público, que sea imposible relacionar los pacientes con sus resonancias. Se pueden ver más ejemplos de este tipo de imágenes suministradas en la Figura 3.

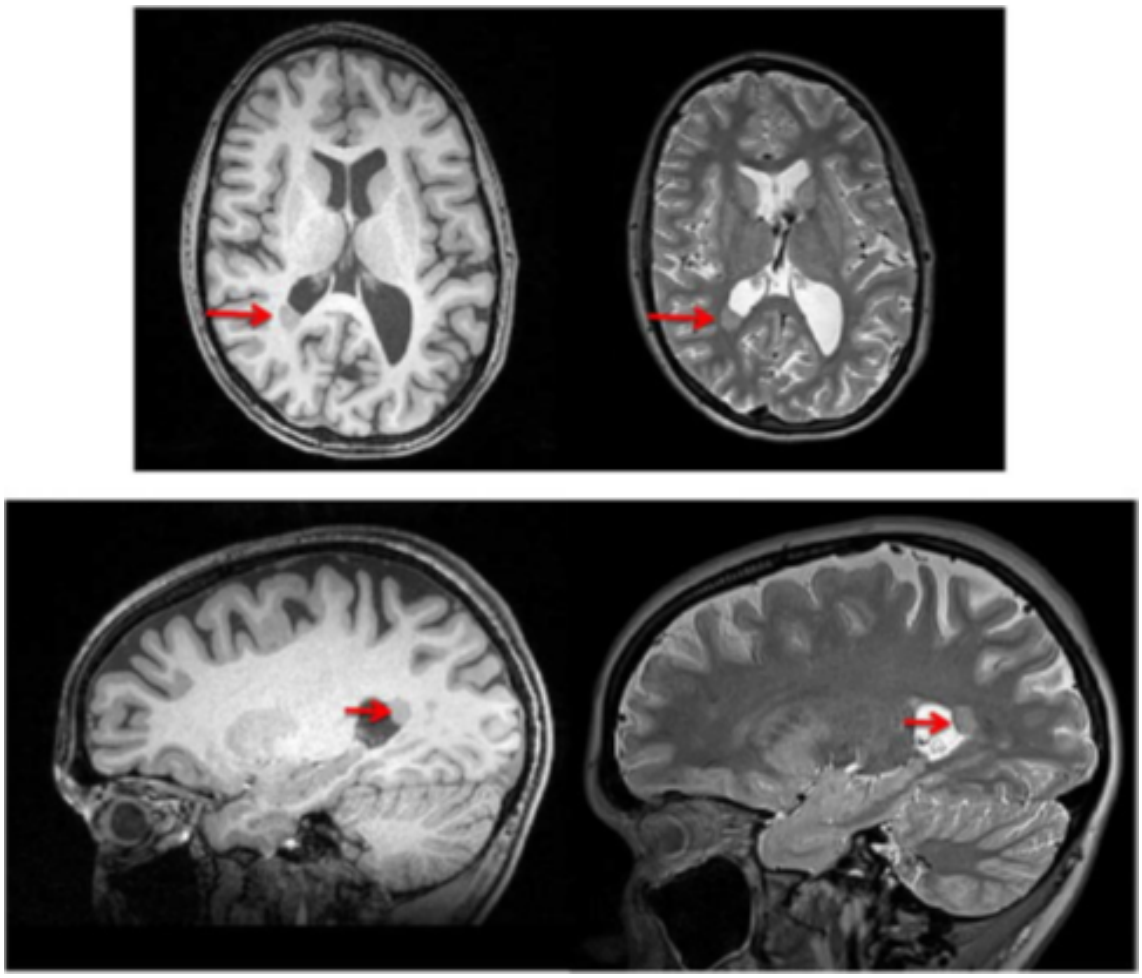

Figura 3: Imágenes axiales y sagitales potenciadas en T1 (izquierda) y T2 (derecha). [9]

1.5. Objetivos

La principal motivación de este proyecto es ayudar en la detección de displasias relacionadas con la epilepsia. Este trabajo complementa la experiencia del experto con la intención de apoyarle en una identificación más ágil. Cuando se quiere localizar una displasia dentro de una imagen se debe tener una formación específica y contar con una cierta experiencia en este sector, ya que no todos los doctores están especializados en este tipo de alteraciones.

Si un ordenador fuese capaz de detectar estas displasias corticales, facilitaría la actividad de los expertos, ayudándoles en la identificación de estas y ahorrándoles tiempo en su labor.

Por todo ello, nuestros objetivos principales en este proyecto serán:

- Utilizar técnicas de visión por computador para realizar el preprocesamiento de las imágenes.
- Utilizar técnicas de Machine Learning para desarrollar un detector preciso, que sea capaz de clasificar las imágenes como “Displasia” o “No displasia”, en función de la aparición de esta.

2
Métodos

En este capítulo, se tratarán los aspectos más teóricos del trabajo: los procesos para la minería de datos y las redes neuronales convolucionales.

Un proyecto de este tipo en el que se tiene una fecha límite de entrega no puede tomarse a la ligera y sin seguir una serie de pasos, por lo que hablaremos de las diferentes metodologías existentes y nos centraremos en la CRISP-DM que es la que hemos empleado para la realización de este trabajo.

También se explicará el funcionamiento de las redes neuronales convolucionales, así como el funcionamiento de la red EfficientNet que es la red elegida para la realización de nuestro detector.

2.1. CRISP-DM

Para este proyecto usaremos la metodología CRISP-DM que es una de las más comunes dentro de la minería de datos.

CRISP-DM cuyas siglas significan *Cross-Industry Standard Process for Data Mining*, es una metodología empleada para orientar los trabajos de minería de datos:

- Como metodología, incluye descripciones de las fases normales de un proyecto, las tareas necesarias en cada fase y una explicación de las relaciones entre las tareas.
- Como modelo de proceso, CRISP-DM ofrece un resumen del ciclo vital de minería de datos.

El ciclo vital contiene seis fases, pero la secuencia de las fases no es estricta, la mayoría de los proyectos avanzan y retroceden entre fases si es necesario. Además, CRISP-DM es flexible y se puede personalizar fácilmente para adaptarse a las necesidades de la organización.

2.1.1. Fases CRISP-DM

El plan de trabajo que se va a llevar a cabo es la metodología CRISP-DM que consta de 6 fases como se puede observar en la Figura 4:

1. Definición de necesidades del cliente: Esta fase inicial se enfoca en la comprensión de los objetivos de proyecto.

2. Estudio y comprensión de los datos: Comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre los datos, y/o descubrir subconjuntos interesantes para formar hipótesis.

3. Análisis de los datos y selección de características: Cubre todas las actividades necesarias para construir el conjunto final de datos a partir de los datos iniciales. Se pueden emplear tablas y registros, así como la transformación de los datos para la realización del modelizado.

4. Modelizado: En esta fase, se seleccionan y aplican las técnicas de modelizado que sean pertinentes al problema y se calibran sus parámetros a valores óptimos. Existen diversos modelos para los problemas de minería de datos, muchos de ellos necesitarán de un tratamiento específico de datos, es por ello que muchas veces se acaba volviendo al proceso de análisis de datos.

5. Evaluación: En esta etapa del proyecto, se han construido uno o varios modelos que parecen alcanzar la calidad suficiente desde una perspectiva de análisis de datos, serán evaluados por los expertos para ver si se alcanzan los objetivos de la primera etapa y se decidirá cuál de ellos es el más adecuado.

6. Uso: El conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo.

2.2. Redes Neuronales Convolucionales

A lo largo de este apartado vamos a tratar de explicar el funcionamiento de las redes neuronales convolucionales.

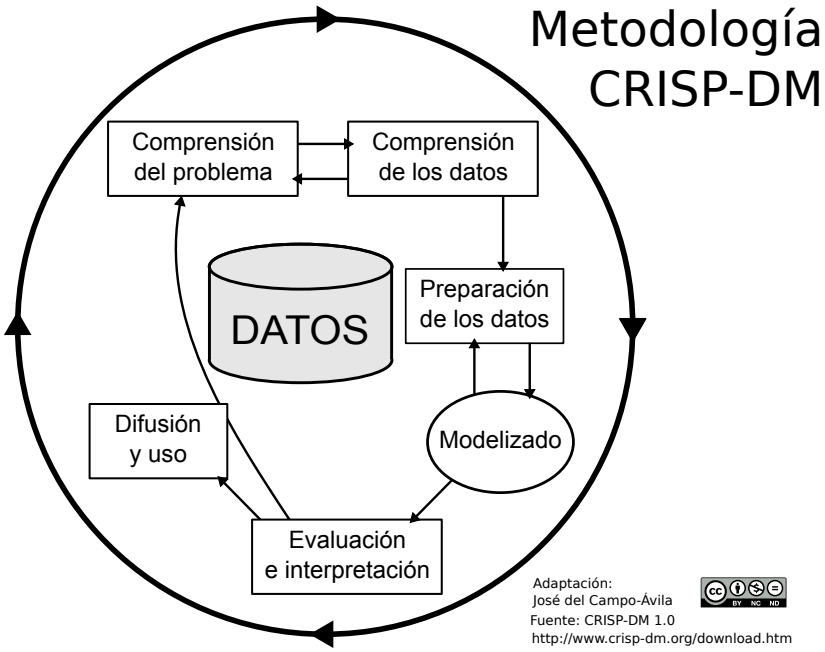

Figura 4: Fases metodología CRISP-DM [3]

Como comentamos previamente, las CNN son un tipo de red neuronal artificial que tienen la capacidad de identificar las distintas características de sus entradas, permiten detectar objetos y "ver" como si de un ojo humano se tratase.

Sus primeras capas pueden detectar líneas, curvas y las más profundas se van especializando hasta poder reconocer formas complejas como un rostro, siluetas, etc.

Para que estas redes sean capaces de realizar lo expuesto anteriormente se necesita seguir una serie de pasos que serán explicados a continuación.

2.2.1. Preprocesamiento

La red neuronal deberá aprender por sí sola a reconocer una diversidad de objetos dentro de imágenes y para ello se necesitará una gran cantidad de imágenes para que la red pueda captar sus características únicas de cada objeto y a su vez, poder generalizarlo.

La red toma como entrada los píxeles de una imagen. Las imágenes tendrán un tamaño de DxD píxeles, donde D es la dimensión de un lado del cuadrado, pero para poder hablar de todo el proceso de manera más cómoda, vamos a tomar el ejemplo de una imagen de 28x28 píxeles de alto y ancho, eso equivaldría a 784 neuronas, y eso sería si solo tenemos un color (escala de

grises).

Antes de alimentar la red, debemos normalizar los valores de la entrada. Los colores de los píxeles van de 0 a 255, por tanto, haremos una transformación de cada píxel: Valor/255, esto nos dará un número entre 0 y 1. Se puede observar un ejemplo de ello en la Figura 5

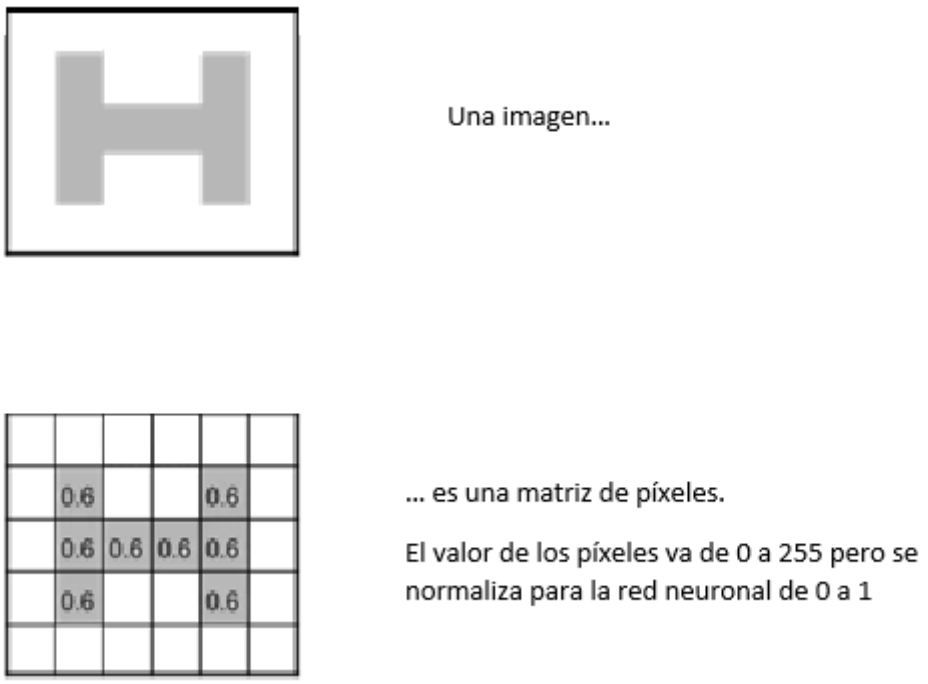

Figura 5: Preprocesamiento de una imagen en las Redes Neuronales Convolucionales. Adaptado de [4]

2.2.2. Convoluciones

Es el proceso distintivo de las CNN. Las convoluciones consisten en tomar grupos de píxeles cercanos de la imagen de entrada e ir operando matemáticamente, mediante el producto escalar, contra una pequeña matriz que se llama kernel. Ese kernel supongamos de tamaño 3x3 píxeles recorre todas las neuronas de entrada, siempre de izquierda a derecha y de arriba abajo, y genera una nueva matriz de salida, que en definitiva será nuestra nueva capa de neuronas ocultas.

El kernel tomará inicialmente valores aleatorios y se irán ajustando mediante backpropagation. Una mejora es hacer que siga una distribución normal siguiendo simetrías, pero sus valores son aleatorios. Se muestra en la Figura 6 un ejemplo de kernel para la imagen anterior.

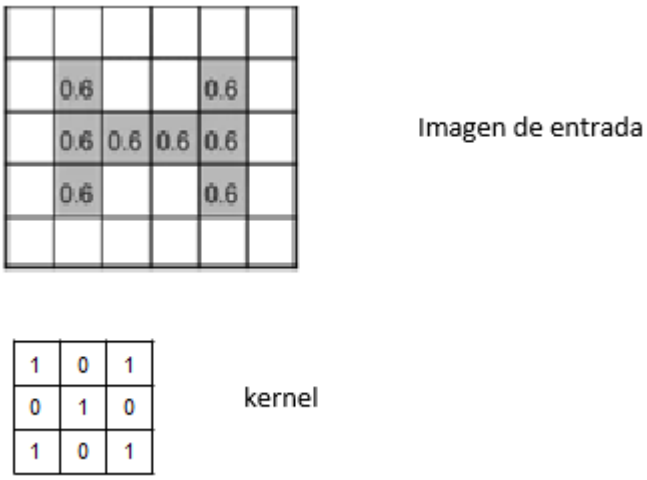

Figura 6: Ejemplo de imagen de entrada y kernel empleado en la convolución. Adaptado de [4]

En realidad, no solo aplicaremos 1 único kernel, si no que tendremos muchos kernels, su conjunto se denomina filtros. Por ejemplo, en una primera convolución podríamos tener 32 filtros, por lo que tendríamos 32 matrices de salida, a este conjunto se le conoce como "*feature mapping*", cada una de 28x28x1 (ya que son imágenes de un solo color) dando un total de 25.088 neuronas para nuestra primera capa oculta de neuronas. A continuación se puede observar en la Figura 7 el proceso de obtención del feature mapping.

Como podemos observar se realiza un producto matricial entre el kernel y la imagen de entrada, se va desplazando de un en un píxel de izquierda a derecha y de arriba a abajo y va generando una nueva matriz que compone al mapa de características.

A medida que vamos desplazando el kernel obtenemos una "nueva imagen" filtrada por el kernel. En esta primera convolución y siguiendo el ejemplo anterior, es como si obtuviéramos 32 "imágenes filtradas nuevas". Estas imágenes nuevas lo que están es "dibujando" ciertas características de la imagen original.

2.2.3. Relleno

Es una operación que se puede usar en las redes convolucionales. El relleno (*padding*) se aplica agregando píxeles de valor cero alrededor de la imagen original. Un ejemplo de aplicar padding a una imagen es la Figura 8.

Tiene dos usos:

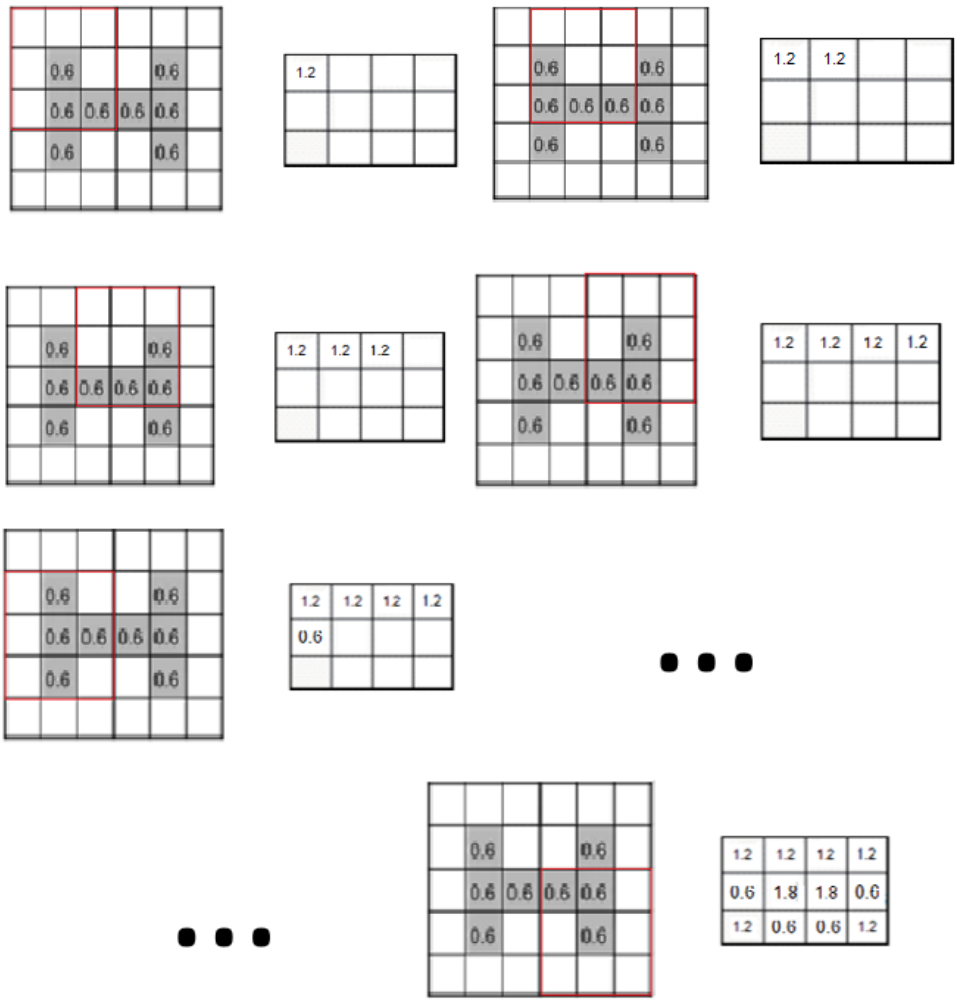

Figura 7: Proceso de obtención del feature mapping. Adaptado de [4]

- El primero es para que al realizar la convolución la imagen resultante sea de igual tamaño que la imagen original.
- El segundo se produce cuando se tiene información relevante en las esquinas de la imagen por lo que al realizar convolución el filtro pasa más por el centro de la imagen que en las esquinas, por lo que se aplica el padding para tener la información más relevante cerca del centro.

2.2.4. Función de activación ReLu

La función más utilizada para este tipo de redes neuronales es la llamada ReLu por *Rectifier Linear Unit* y consiste en

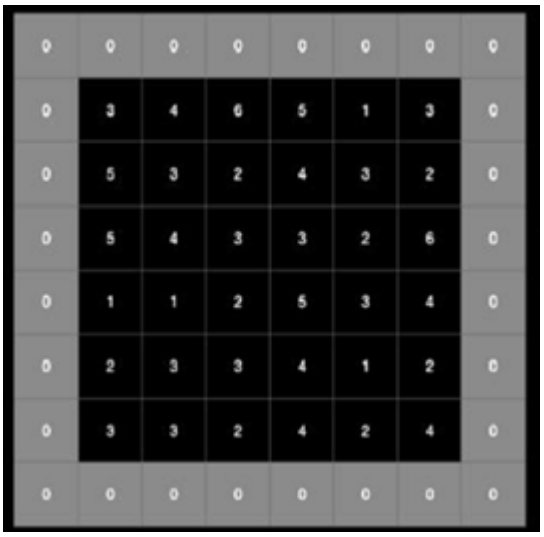

Figura 8: Ejemplo de aplicación de Padding. [8]

$$f(x) = max(0, x)$$

Cuando procesamos una imagen, cada capa de convolución debe capturar algún patrón en la imagen y pasarla en la siguiente capa de convolución. Los valores negativos no son importantes en el procesamiento de imágenes y, por lo tanto, se establecen en 0. Pero los valores positivos después de la convolución deben pasar a la siguiente capa. Es por eso que ReLu se utiliza como función de activación. Si utilizamos sigmoide o tanh, la información se pierde ya que ambas funciones modificarán las entradas a un rango muy cerrado. En la Figura 9 se puede observar la gráfica de la función ReLu.

2.2.5. Submuestreo

Como hemos podido observar anteriormente el número de neuronas es muy grande, ya que hay una por cada píxel de la imagen, si a esto le sumamos que ese número aumenta considerablemente cada vez que obtenemos una capa oculta, el número de neuronas se volvería incontrolable e implicaría un mayor procesamiento.

Para reducir el tamaño de la próxima capa de neuronas haremos un proceso de submuestreo (*subsampling*) en el que reduciremos el tamaño de nuestras imágenes filtradas, pero en donde deberán prevalecer las características más importantes que detectó cada filtro. Hay diversos tipos de subsampling, pero el más usado es el Max-Pooling. Para este proyecto como

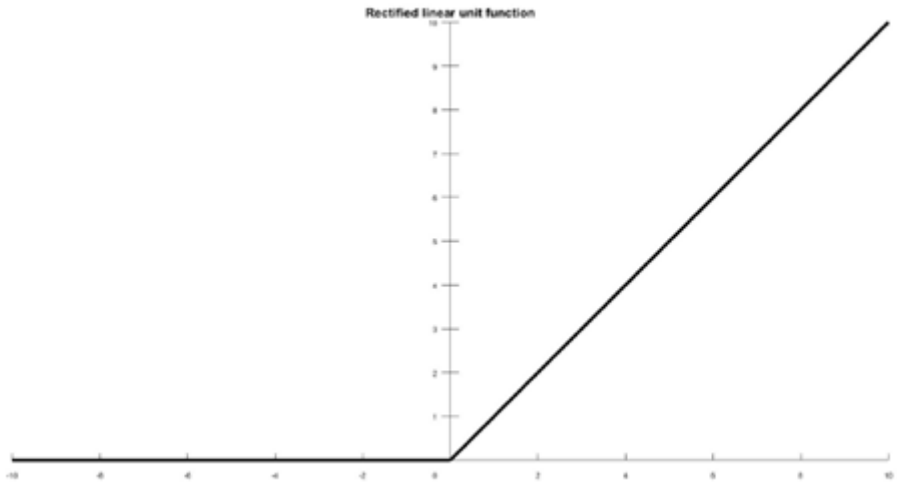

Figura 9: Función de activación ReLu. [6]

hemos empleado una EfficientNet usaremos diferentes métodos de subsampling como Batch-Normalization, Reshape o Conv2D, pero entre ellos vamos a destacar Global Average Pooling 2D, a continuación, vamos a hablar de él para seguir con nuestro ejemplo de la imagen de 28x28 píxeles.

Global Average Pooling 2D representa una operación de agrupación promedio. Este bloque genera un tensor más pequeño que su entrada, lo que significa que los bloques descendentes necesitarán menos parámetros y cantidad de cálculo; también sirve para controlar el sobre-ajuste.

El bloque de Agrupación de Promedio 2D mueve un rectángulo (ventana) sobre los datos entrantes, calculando el promedio en cada ventana específica. El tamaño de la ventana está determinado por el factor de agrupación horizontal y vertical y los pasos grandes que da la ventana están determinados por el paso horizontal y vertical.

Los bloques de agrupación promedio se insertan después de uno o más bloques convolucionales; ayudan al bloque convolucional interno a recibir información de una porción más grande de la imagen original (un filtro de 3x3 después de la agrupación está influenciado por una porción de 6x6 del tensor antes de la agrupación). Si vemos bloques convolucionales como detectores de una característica específica, la agrupación promedio encuentra el valor ”medio”de esa característica dentro del rectángulo de agrupación. Cada canal (por lo tanto, cada función) se trata por separado. Se puede observar en la Figura 10 un ejemplo de aplicación del

Global Average Pooling 2D.

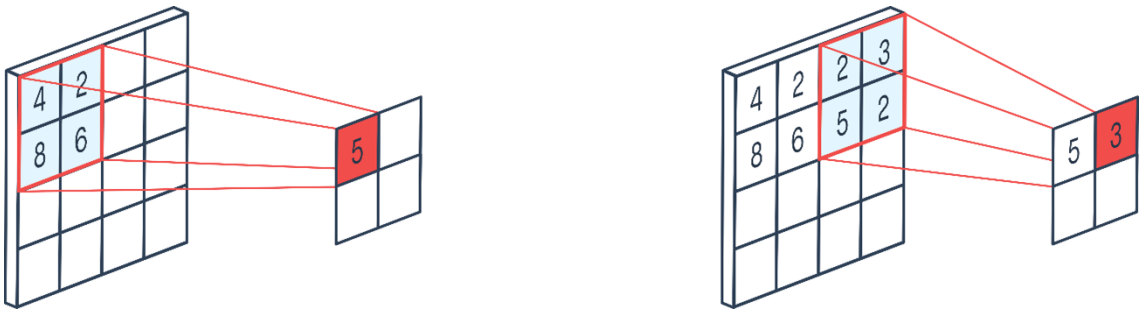

Figura 10: Ejemplo de los primeros pasos de Global Average Pooling 2D. [1]

Continuando pues con nuestro ejemplo de 28x28 píxeles en imágenes de un solo color, aplicándole un kernel de 32 filtros y usando Global Average Pooling 2D con un factor de agrupación horizontal y vertical de 2, obtendremos una salida de la convolución de 14x14x32. Se puede ver un ejemplo de nuestra primera convolución en la Figura 11.

1)Imagen	2)Aplica Kernel	3)Feature Mapping	4)Global Average Pooling 2D	5)Salida
28x28x1	32 filtros de 3×3	28x28x32	de 2×2	14x14x32

La primera convolución es capaz de detectar características primitivas como líneas o curvas. A medida que hagamos más capas con las convoluciones, los mapas de características serán capaces de reconocer formas más complejas, y el conjunto total de capas de convoluciones podrá "ver". En la Figura 12 aparecerá la segunda convolución de nuestra CNN.

1)Imagen	2)Aplica Kernel	3)Feature Mapping	4)Global Average Pooling 2D	5)Salida
14x14x32	64 filtros de 3×3	14x14x64	de 2×2	14x14x32

La tercera convolución comenzará en tamaño 7×7 pixeles y luego del Average Pooling 2D quedará en 3×3 con lo cual podríamos hacer sólo 1 convolución más. En este ejemplo empezamos con una imagen de 28x28 píxeles e hicimos 3 convoluciones. Si la imagen inicial hubiese sido mayor aún hubiéramos podido seguir haciendo convoluciones.

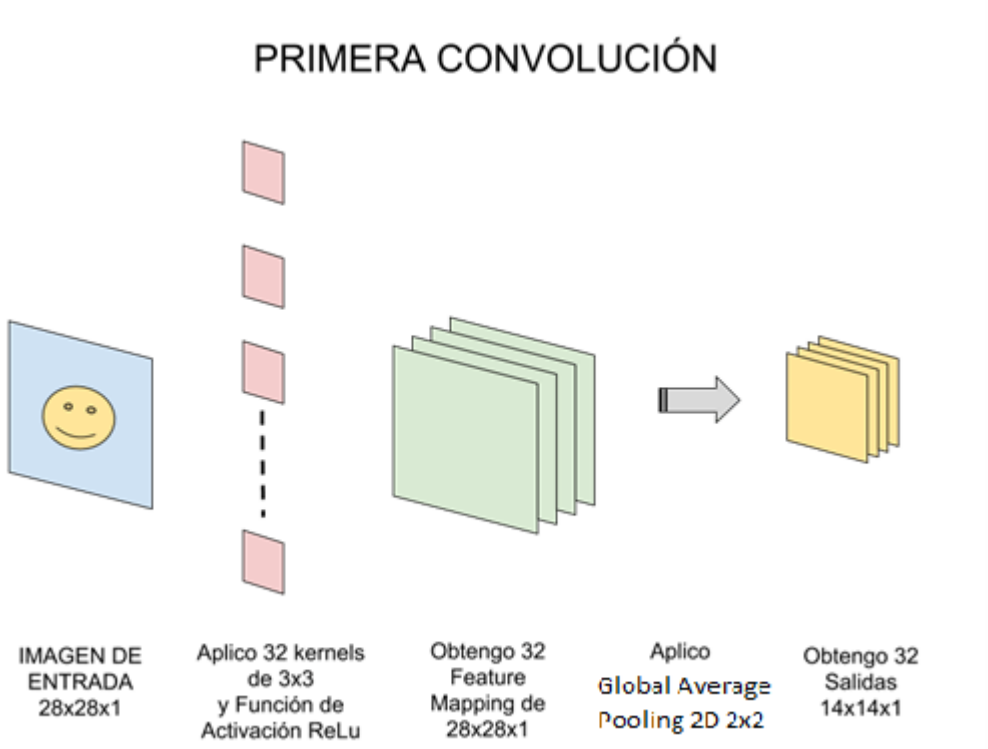

Figura 11: Imagen de la primera Convolución de nuestra CNN. Adaptado de [4]

1)Imagen	2)Aplica Kernel	3)Feature Mapping	4)Global Average Pooling 2D	5)Salida
7x7x64	128 filtros de 3×3	7x7x128	de 2×2	3x3x128

2.2.6. Conectar con una red neuronal tradicional y Transfer Learning

Tomaremos la última capa oculta a la que hicimos subsampling, que se dice que es tridimensional por tomar la forma (alto ancho, mapas) y la aplanamos, esto deja de ser tridimensional y pasa a ser una capa de neuronas tradicionales. Se puede ver la Arquitectura de una CNN en la Figura 13.

A esta nueva capa oculta tradicional, le aplicamos una función llamada Softmax que conecta contra la capa de salida final que tendrá la cantidad de neuronas correspondientes con las clases que estamos clasificando. Si clasificamos perros, gatos y pájaros serán 3 neuronas, en nuestro caso van a ser 2, Displasia y No Displasia.

Las salidas tendrán formato conocido como "*one-hot-encoding*" en el que para Displasia y

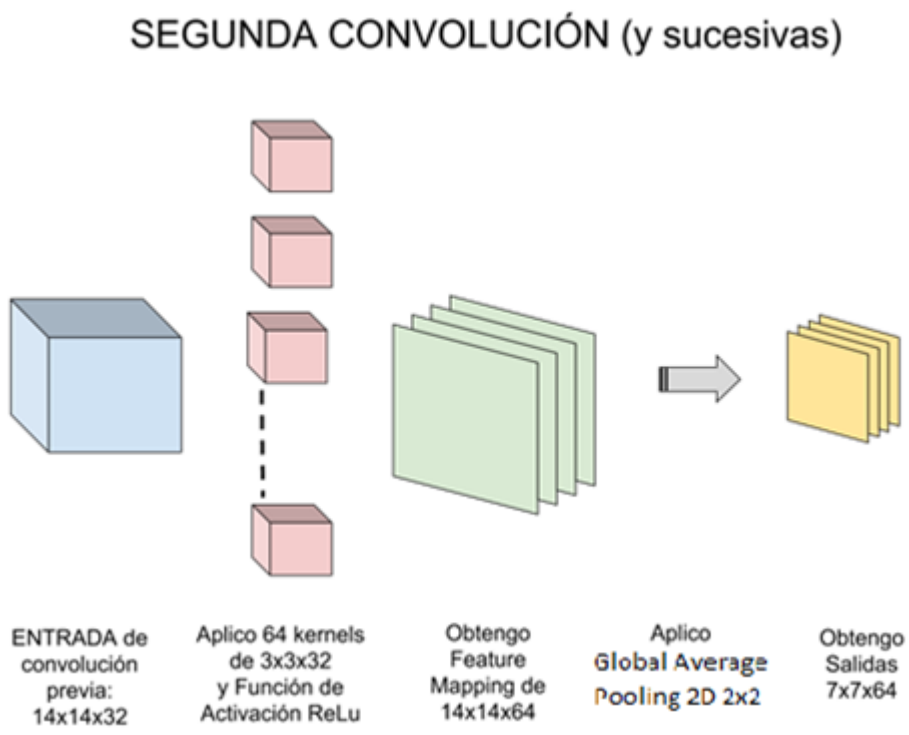

Figura 12: Imagen de la segunda Convolución de nuestra CNN. Adaptado de [4]

No Displasia será [1,0] y [0,1].

La función Softmax se encarga de pasar a probabilidad las neuronas de salida, suele usarse cuando nos encontramos en una tarea de clasificación con dos o más clases.

La función toma la entrada del vector de C números reales (donde C es el número de clases que se quiere que la red clasifique) y lo normaliza en una distribución de probabilidad que consiste en C probabilidades, que será la predicción que haya realizado la red. El máximo porcentaje será para la clase a la que es más probable que pertenezca la imagen de entrada.

Para ello empleará la siguiente fórmula.

$$f_i(x) = \frac{exp(x_i)}{\sum_{j=1}^{C} exp(x_j)}$$

Por ejemplo, una salida [0,6 0,4] nos indicaría que hay un 60 % de que haya displasia y un 40 % de que no.

Todo lo explicado previamente sería si nosotros mismos montáramos nuestra propia CNN, pero normalmente para este tipo de proyectos se usa lo que se llama *Transfer Learning*.

El transfer learning es una de las técnicas más importantes del Deep Learning para el

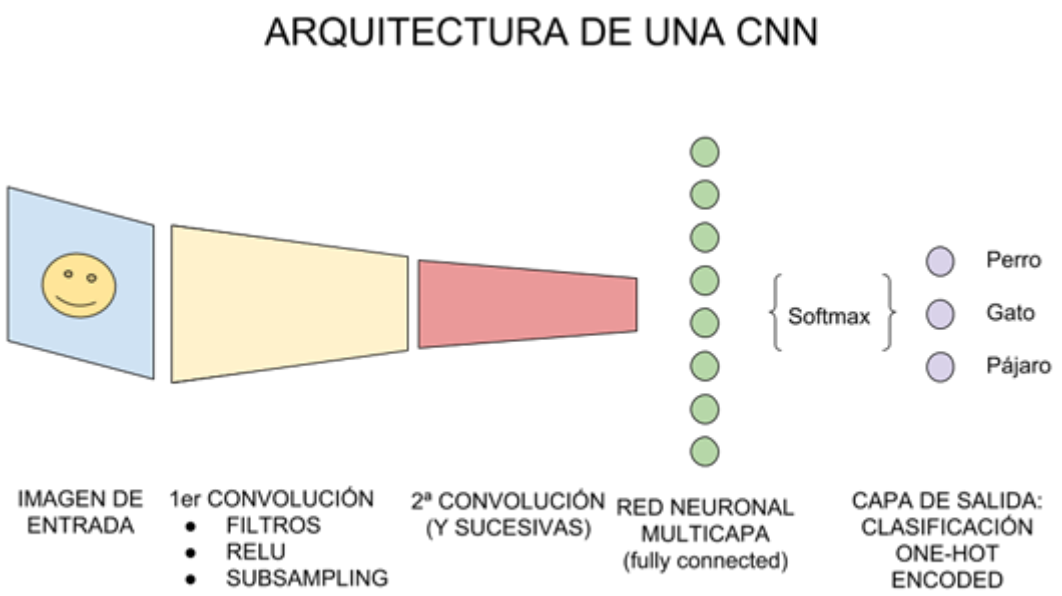

Figura 13: Arquitectura de una CNN. [4]

aprendizaje automático en inteligencia artificial. Consiste, en esencia, en aprovechar una gran cantidad de información relacionada con la resolución de un problema y utilizarla sobre otro distinto, pero que comparta ciertas características con este. Modifica redes neuronales ya entrenadas para reconocer ciertas características, y así poder reconocer otras similares.

Las capas convolucionales son filtros que se aplican a las imágenes, y son básicamente las encargadas de la extracción de características. Cuanto más profunda sea en la red, más específica es la función extraída.

La última capa es básicamente una red clásica que puede ser menos costosa de entrenar. Se encarga de tomar la salida de la primera parte y clasificarla.

El aprendizaje de transferencia es una técnica en la que tomamos una red pre-entrenada, eliminamos la última parte de la red (el clasificador) y la reemplazamos con la nuestra. Luego entrenamos a nuestro clasificador. Esto nos permite usar arquitecturas menos costosas.

2.2.7. Backpropagation

El proceso es similar al de las redes tradicionales en las que tenemos una entrada y una salida esperada y mediante el *backpropagation* mejoramos el valor de los pesos de las interconexiones entre capas de neuronas y a medida que iteramos esos pesos se ajustan hasta ser

óptimos.

En el caso de la CNN, debemos ajustar el valor de los pesos de los distintos kernels. Esto es una gran ventaja en el aprendizaje, ya que como hemos visto el kernel es de un tamaño reducido. Si se ajustarán como las neuronas tradicionales, el número de ajustes sería muchísimo mayor.

2.2.8. Métricas

Previamente se ha podido observar cómo se crea nuestra red neuronal convolucional, para este apartado hablaremos sobre las métricas que emplearemos en este trabajo para determinar cómo de bueno es un modelo.

- Cruce de entropía categórica (Categorical crossentropy): Es una función de pérdida que se utiliza en tareas de clasificación de clases múltiples. Estas son tareas en las que un ejemplo solo puede pertenecer a una de las categorías posibles, y el modelo debe decidir cuál. Formalmente, está diseñado para cuantificar la diferencia entre dos distribuciones de probabilidad. Softmax es la única función de activación recomendada para usar con la función de pérdida de entropía cruzada categórica.

- Precisión: Con la métrica de precisión podemos medir la calidad del modelo de machine learning en tareas de clasificación. La fórmula para obtener la precisión es la siguiente:

$$Precision = \frac{TP + TN}{TP + TN + FP + FN}$$

2.3. Arquitectura empleada

Como hemos comentado previamente, para la realización de nuestro clasificador necesitaremos una red neuronal convolucional ya entrenada, la cual solo cambiaremos su última capa para obtener los resultados esperados acerca de si las imágenes poseen o no displasia.

Existen numerosas redes neuronales convolucionales ya entrenadas y que se utilizan para multitud de ámbitos, como pueden ser las redes ResNet, SENet o Inception.

Para este proyecto no vamos a contar con tantas imágenes como podrían necesitarse en este tipo de trabajo, ya que nuestras imágenes no están en ningún banco de datos, sino que son aquellas que nos pueden facilitar desde el Hospital Regional Universitario de Málaga. Nos

hemos centrado en la búsqueda de una red que nos garantice una gran eficiencia y precisión para este reducido número de imágenes.

Las redes neuronales convolucionales (CNN) se desarrollan comúnmente con un coste de recursos fijo y luego se amplían para lograr una mayor precisión cuando se ponen a disposición más recursos. Por ejemplo, ResNet se puede escalar de ResNet-18 a ResNet-200 aumentando el número de capas. La práctica convencional para el escalado de modelos es aumentar arbitrariamente la profundidad o el ancho de CNN, o utilizar una resolución de imagen de entrada más grande para entrenamiento y test. Si bien estos métodos mejoran la precisión, generalmente requieren un tedioso ajuste manual y, a menudo, su rendimiento no es óptimo. Es por todo esto que los expertos han creado un método basado en principios para escalar una CNN y así obtener una mayor precisión y eficiencia.

2.3.1. Dimensiones de escala

Hay tres dimensiones de escala de una CNN: profundidad, ancho y resolución. Para comprender el efecto de escalar la red, estudiamos sistemáticamente el impacto de escalar diferentes dimensiones del modelo. Si bien escalar dimensiones individuales mejora el rendimiento del modelo, se puede observar que equilibrar todas las dimensiones de la red (ancho, profundidad y resolución de imagen) con los recursos disponibles mejoraría el rendimiento general. En la Figura 14 se muestra las diferentes escalas del modelo.

- La profundidad (d) nos habla acerca de cómo de profunda es la red, lo que equivale a la cantidad de capas que contiene. Es la forma más común de escalar. La profundidad se puede aumentar y reducir agregando y eliminando capas respectivamente. Escalar en profundidad nos permite que cuantas más profunda sea la red, esta podrá capturar características más ricas y complejas. Con más capas en teoría el rendimiento debería mejorar, pero en la práctica esto no siempre sucede. La desaparición de gradientes es uno de los problemas más comunes que surgen a medida que profundizamos, otro problema se produce cuando agregamos demasiadas capas ya que no siempre ayuda. Un ejemplo de esto es la red ResNet-1000 que tiene una precisión similar a ResNet-101.

- Ancho (w) simplemente significa cómo de ancha es la red. Una medida de ancho, por ejemplo, es la cantidad de canales en una capa de convolución. La escala de ancho se

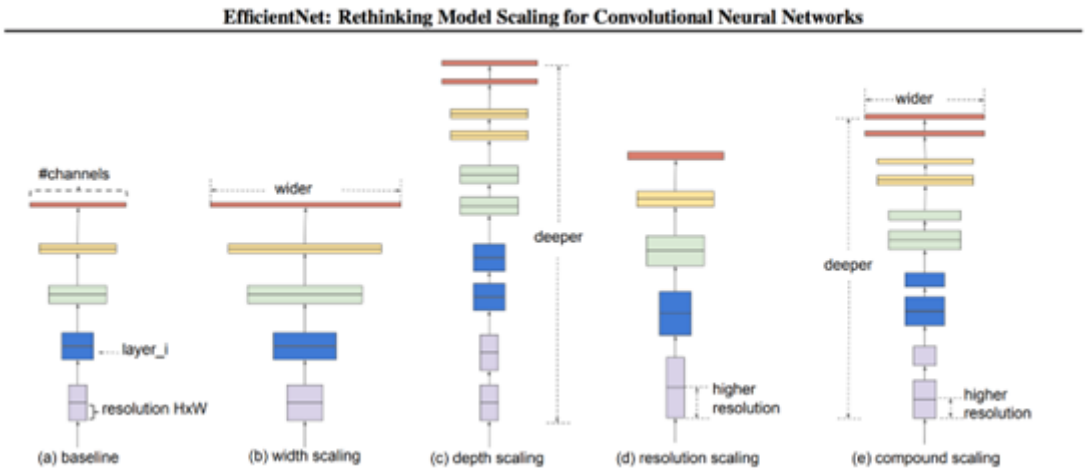

Figura 14: Escala del modelo. (a) es un ejemplo de red de referencia; (b) - (d) son escalas convencionales que solo aumentan una dimensión del ancho, la profundidad o la resolución de la red. (e) Método de escala compuesto propuesto que escala uniformemente las tres dimensiones con una proporción fija. [5]

usa cuando queremos mantener nuestro modelo pequeño. Las redes más amplias tienden a poder capturar características más detalladas. Además, los modelos más pequeños son los más fáciles de entrenar. Por otro lado, con modelos menos profundos, pero más amplios la precisión se satura rápidamente con un ancho mayor.

- Cuando hablamos de resolución nos referimos a la resolución de las imágenes que se pasan a una CNN. Se podría decir que una imagen de alta resolución debería funcionar mejor que otra con baja resolución ya que sus características son más detalladas. La resolución no se escala linealmente ya que la ganancia de precisión disminuye muy rápidamente. Por ejemplo, aumentar la resolución de 500x500 a 560x560 no produce mejoras significativas.

Por tanto, la ampliación de cualquier dimensión de la red mejora la precisión, pero la ganancia de precisión disminuye para los modelos grandes. Se puede ver en la Figura 15 una gráfica que refleja lo anterior.

Podríamos combinar escalas para diferentes dimensiones, pero hay ciertos problemas como se puede observar en la Figura 16:

- Aunque se puedan escalar dos o tres dimensiones arbitrariamente, el escalado arbitrario

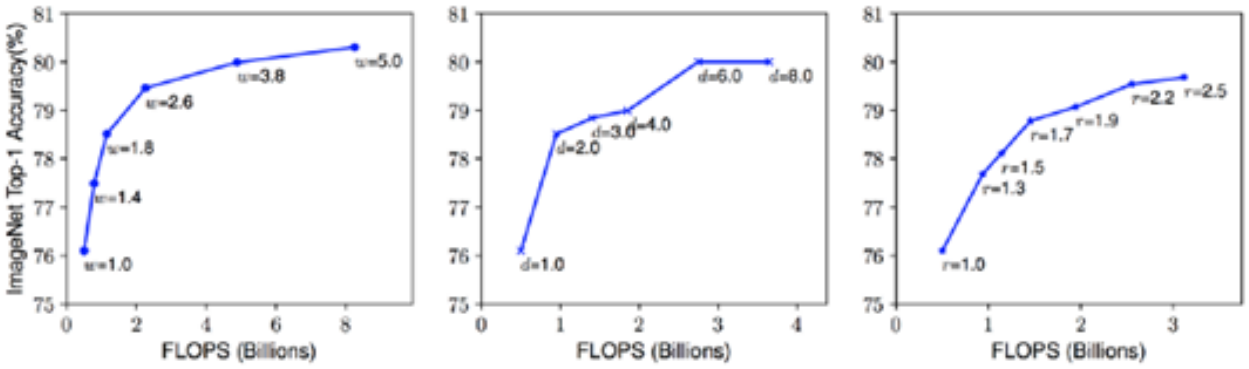

Figura 15: Ampliación de un modelo de línea base con diferentes coeficientes de ancho (w), profundidad (d) y resolución (r) de red. [5]

es algo tedioso.

- El escalado manual da como resultado una precisión y eficacia por debajo de lo esperado.

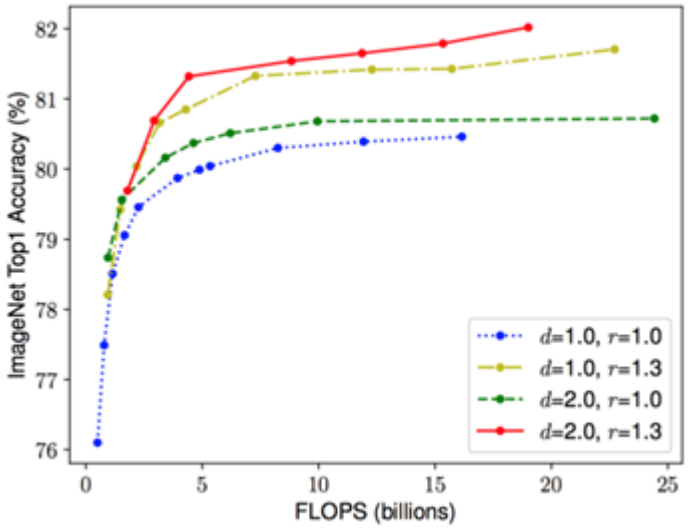

Figura 16: Escalar el ancho de la red para diferentes redes. Cada punto en una línea denota un modelo con diferente coeficiente de ancho (w). [5]

Podemos observar que es fundamental equilibrar todas las dimensiones de una red durante el escalado de la CNN para obtener una mayor precisión y eficiencia.

Entonces para poder realizar el escalado los expertos crearon una escala compuesta propuesta en el que se utilizó el coeficiente compuesto ϕ para escalar uniformemente el ancho, la profundidad y la resolución de la red de una manera basada en:

$$Profundidad : d = \alpha^{\phi}$$

$$Anchura : w = \beta^{\phi}$$

$$Resolucin : r = \gamma^{\phi}$$

$$s.t. \alpha.\beta^2.\gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

ϕ es un coeficiente especificado por el usuario que controla cuántos recursos están disponibles, mientras que α, β y γ especifican cómo asignar estos recursos a la profundidad, el ancho y la resolución de la red, respectivamente.

Los FLOPS (cantidad de operaciones en punto flotante que un sistema puede hacer cada segundo) de una convolución son casi proporcionales a d, w^2, r^2, es decir, duplicar la profundidad duplicará los FLOPS mientras que duplicar el ancho o la resolución aumenta los FLOPS casi cuatro veces. Por tanto, para asegurarse de que el total de FLOPS no exceda de 2^{ϕ} la restricción es que $\alpha.\beta^2.\gamma^2 \approx 2$

2.3.2. Arquitectura EfficientNet

El escalado no cambia las operaciones de la capa, por lo que es ideal tener una buena red de base y luego escalarla a lo largo de diferentes dimensiones utilizando el escalado compuesto propuesto. Los expertos utilizaron como red base la búsqueda de arquitectura neuronal (NAS) que optimiza tanto la precisión como los FLOPS. La arquitectura es similar a M-NASNet. Las capas/bloques de la red se pueden observar en la Figura 17.

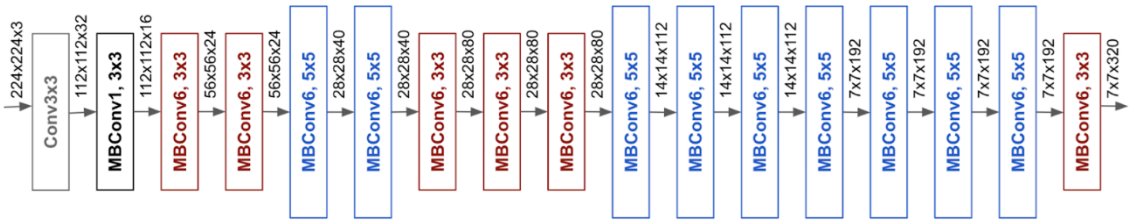

Figura 17: Arquitectura de la red básica EfficientNet-B0. [7]

Ahora que tenemos la red base, vamos a buscar valores óptimos para nuestros parámetros de escala. Tenemos un total de cuatro parámetros que buscar: $\alpha, \beta, \gamma, \phi$. La búsqueda de estos parámetros se puede completar en dos pasos:

- Con $\phi = 1$, asumiendo que hay dos veces más recursos disponibles, y haciendo una búsqueda de cuadrícula pequeña para α, β, γ. Para la red de referencia B0, resultó que los valores óptimos son $\alpha = 1{,}2$, $\beta = 1{,}1$ y $\gamma = 1{,}15$ tales que $\alpha.\beta^2.\gamma^2 \approx 2$.
- Ahora si fijamos α, β, γ como constantes y probamos con diferentes valores para ϕ. Los diferentes valores de ϕ producen los EfficientNets B1-B7.

2.3.3. Rendimiento EfficientNet

Hemos comparado las redes EfficientNets con otras CNN existentes en ImageNet. En general, los modelos EfficientNet logran una mayor precisión y una mejor eficiencia sobre las CNN existentes, reduciendo el tamaño de los parámetros y los FLOPS en un orden de magnitud. Esta investigación ha sido realizada por [7]. En la Figura 18 se muestra una gráfica con la comparativa entre modelos.

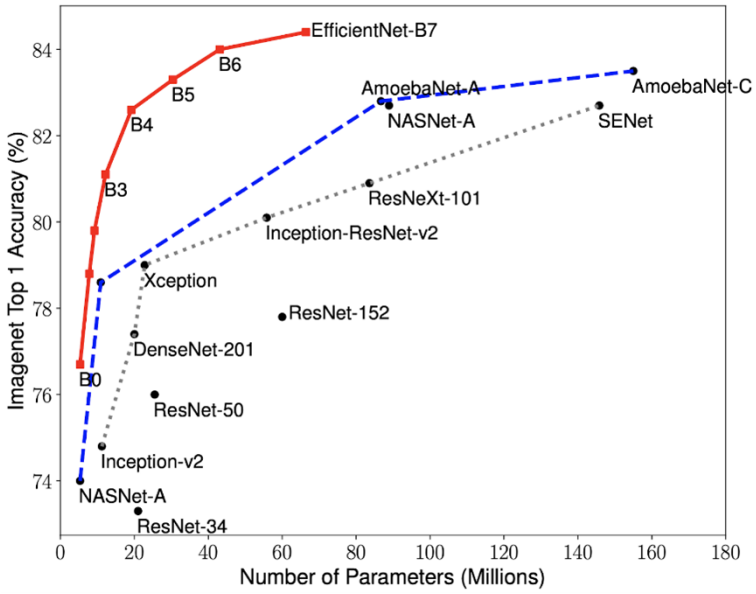

Figura 18: Comparación del tamaño del modelo frente a la precisión. [7]

Aunque las EfficientNets funcionan bien en ImageNet su mayor utilidad está en su empleo dentro de las transferencias de aprendizaje. Se han realizado estudios en los que se han

evaluado las EfficientNets en ocho conjuntos de datos de aprendizaje por transferencia ampliamente utilizados. EfficientNet logró una gran precisión en 5 de los 8 conjuntos de datos, como CIFAR-100 y Flowers.

Debido a su gran precisión, eficacia y su magnífica respuesta a la hora de aplicarla en las transferencias de aprendizaje hemos decidido que la red EfficientNet es una red con unas cualidades idóneas para este proyecto.

3

Aplicación CRISP-DM

Como hablamos anteriormente, CRISP-DM será la metodología empleada para la realización de este proyecto por su estrecha relación con la minería de datos. A lo largo de este capítulo hablaremos del desarrollo de nuestro trabajo y qué se fue averiguando y desarrollando en cada una de las seis fases de esta metodología.

3.1. Definición de necesidades del cliente

Esta fase inicial se enfocará en la comprensión de los objetivos de proyecto.

Esta primera fase de definición de las necesidades del cliente se inició a partir de una reunión con Beatriz Asenjo y Victoria Fernández, doctoras del Hospital Regional Universitario de Málaga del departamento de Radiodiagnosis donde nos exponen las necesidades del centro de crear un detector de displasias corticales que sirva de ayuda al experto a la hora de diagnosticar epilepsia en un paciente, debido a que este tipo de anomalías necesitan un conocimiento que solo estos profesionales poseen.

Para automatizar este tipo de trabajo es necesario aplicar técnicas de minería de datos que nos permitan realizar una correcta detección de estas displasias. Para ello se debe contar con un banco de imágenes con las que poder realizar el estudio, y como un banco de imágenes de este tipo no es algo común, se acordó que deben ser ellos quienes nos pasarían las imágenes de los pacientes. Es por ello por lo que se tuvieron que realizar una serie de documentos de confidencialidad, donde nos comprometemos a no filtrar ningún tipo de datos de los casos facilitados. A lo largo de estos meses se han ido facilitando imágenes que se irán añadiendo al entrenamiento.

3.2. Estudio y comprensión de los datos

Esta segunda fase comienza con la colección de imágenes inicial y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre las imágenes, y/o descubrir subconjuntos interesantes para formar hipótesis en cuanto a la información oculta.

El trabajo encargado por las doctoras necesita de un clasificador que diga si las imágenes introducidas tienen o no displasia para la posterior diagnosis de epilepsia, es por ello por lo que necesitaremos para entrenar un conjunto de imágenes que posean y que no posean dicha anomalía.

Uno de los problemas que han surgido es que la forma en la que las doctoras nos pasaban estas imágenes era sin formato, ya que las aplicaciones que ellas usan para poder ver las resonancias realizan la transformación directamente a DICOM que es el formato más común para este tipo de imágenes. Para poder trabajar con ellas, y como no disponemos de este tipo de visores, se ha decidido hacer una transformación de estos archivos a jpg.

Durante este proyecto hemos usado imágenes Flair y T1, en estos tipos de imágenes se ha podido observar que las displasias tenían una forma muy sutil y apenas reconocible debido a su proporción con respecto a la imagen completa. Por este motivo y ya que no tenemos el conocimiento necesario para saber localizar la displasia, ha sido necesario que las doctoras además de pasarnos las imágenes nos indicaran en qué intervalo de estas estaban dichas anomalías.

Se pudo observar que el número de imágenes en las que aparece la displasia era distinto para cada paciente en función del tamaño de esta. Normalmente para cada caso suele haber un número mucho más pequeño de imágenes que presentan displasia, por tanto, en nuestro banco de imágenes habrá muchas más sin dicha anomalía que con esta, provocando un gran desbalanceo en cuanto al número de unas y de otras.

Para este tipo de trabajo se buscó información acerca del tamaño necesario que deben tener estas imágenes, y basándonos en casos anteriores parecidos a este como puede ser un clasificador de tumores cerebrales [2], se descubrió que lo más común es que todas las imágenes fueran de 224x224 por lo que este fue uno de los primeros pasos dentro del preprocesamiento que sacamos en claro.

3.3. Análisis de los datos y selección de características

En esta fase se cubren todas las actividades necesarias para construir el conjunto final de datos a partir de los datos en bruto iniciales.

Finalmente, para este proyecto hemos contado con 16 casos de pacientes de los que se han conseguido extraer 272 imágenes con displasia y 1161 imágenes sin displasia.

Para este trabajo tendremos dos conjuntos de datos:

- Entrenamiento: Separado en dos subconjuntos (*Displasia* y *No Displasia*).
- Test: Separado en dos subconjuntos (*Displasia* y *No Displasia*).

No se ha realizado ningún proceso de preprocesamiento salvo escalar la imagen a 224x224 para que tratemos con imágenes con el mismo tamaño. En futuras iteraciones del método CRISP-DM no se descarta realizar algún tipo de preprocesamiento, esto último quedará reflejado en las conclusiones.

3.4. Modelizado

En esta fase, se seleccionan y aplican las técnicas de modelizado que sean pertinentes al problema y se calibran sus parámetros a valores óptimos.

Para seleccionar las imágenes de entrenamiento y test existen numerosos estrategias. Nos hemos centrado en tres métodos para quedarnos con el que mayor precisión tenga para usarlo como base a la hora de generar el modelo definitivo:

- Método aleatorio: Selección aleatoria de imágenes tanto para entrenamiento como para test. El 80 % de las imágenes con displasia han ido para entrenamiento y el restante para test. Por otro lado, debido a que el número de imágenes sin displasia era mucho mayor que las que sí contienen esta anomalía, se ha decidido que un 30 % de las mismas vayan para entrenamiento y que el conjunto de test de estas esté formado por el mismo número de imagen que el de test de displasia. La cantidad de imágenes de cada conjunto se puede observar en la Figura 19.
- Método Leave One Out: Todas las imágenes de los pacientes serán usados tanto de entrenamiento de displasia como no displasia, salvo las de uno de ellos que serán empleadas

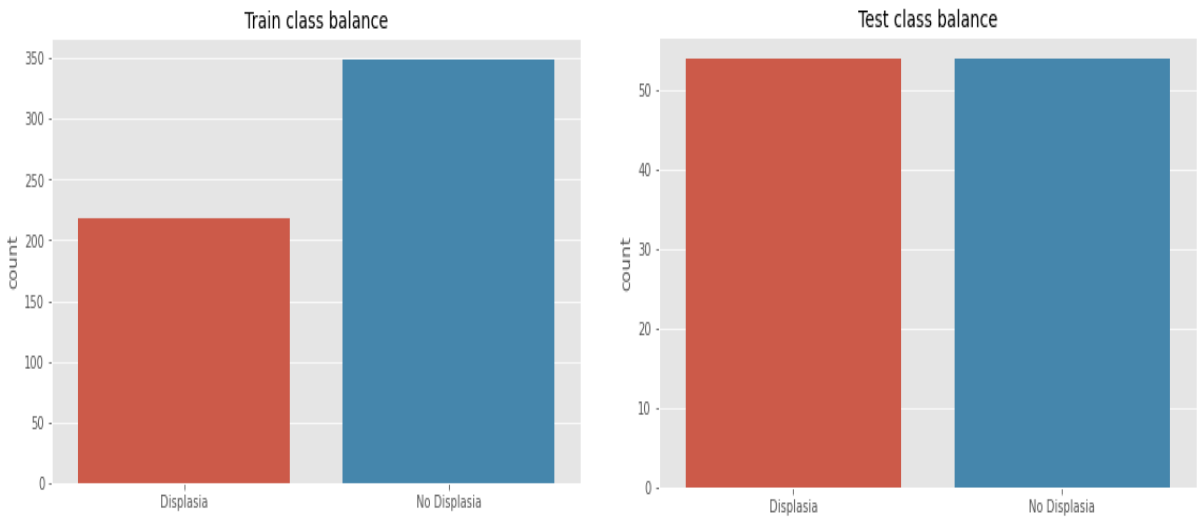

Figura 19: Cantidad de imágenes en el conjunto de entrenamiento (80 % de las imágenes con displasia y 30 % de las imágenes sin displasia) y Test (20 % de las imágenes con displasia y el mismo número de imágenes sin displasia) Método Aleatorio.

como test de displasia y no displasia. Por otro lado, se ha decidido que el 30 % de las imágenes sin displasia de estos pacientes vayan para entrenamiento y que el conjunto de test de estas esté formado por el mismo número de imagenes que el de test de displasia. La cantidad de imágenes de cada conjunto se puede observar en la Figura 20.

- Método de las más próximas: Este método es una propuesta personal que hemos realizado. Utilizamos como conjunto de entrenamiento y de test de Displasia imágenes seleccionadas aleatoriamente, mientras que de test para el conjunto de las No Displasia serán aquellas que estén más próximas a las imágenes con displasia. Por último, destacar que el conjunto de entrenamiento de las imágenes sin esta anomalía estará formado, de manera aleatoria, por el 30 % del total de estas. La cantidad de imágenes de cada conjunto se puede observar en la Figura 21.

Se ha realizado 10 entrenamientos con cada método, para ello se ha cambiado la semilla empleada en cada selección aleatoria, y para el caso del Leave One Out se ha cambiado el paciente.

Se ha creado una matriz para representar qué imágenes Flair y T1 se han usado para tareas de entrenamiento o test para cada uno de los métodos para exponerlo a los expertos y facilitar la evaluación de cada método. Debido a que no podemos mostrar las matrices enteras por falta

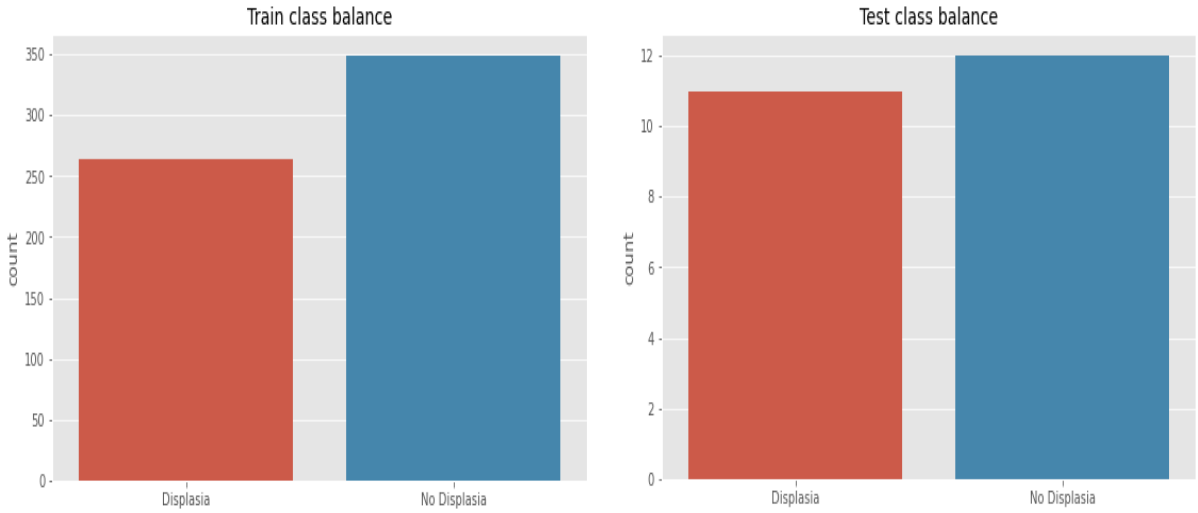

Figura 20: Cantidad de imágenes en el conjunto de entrenamiento (imágenes con displasia de todos los pacientes salvo un caso y 30 % de las imágenes sin displasia) y Test (imágenes con displasia del paciente que se ha dejado fuera y el mismo número de imágenes sin displasia del mismo paciente) Método Leave One Out.

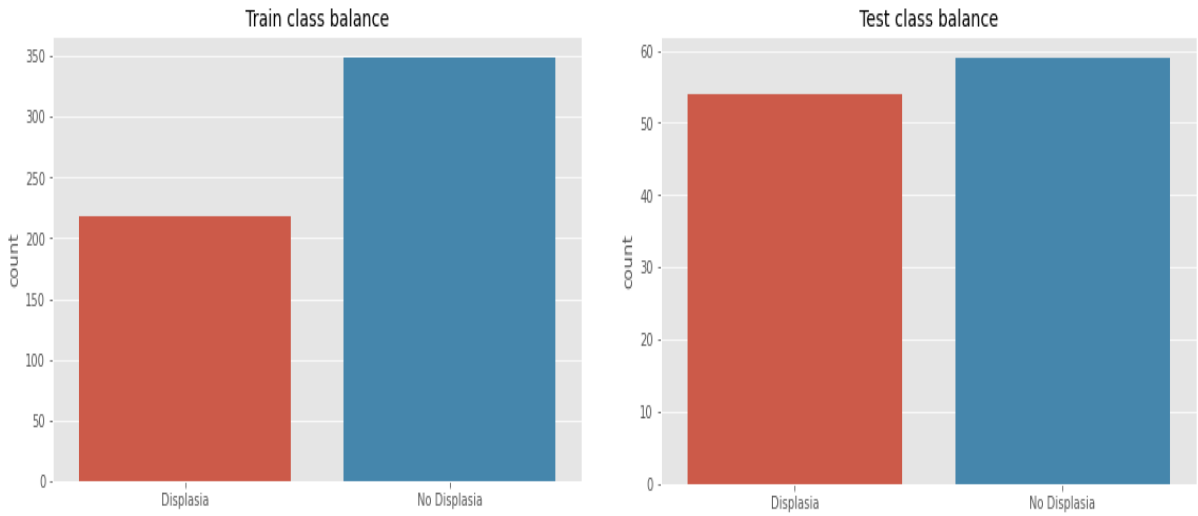

Figura 21: Cantidad de imágenes en el conjunto de entrenamiento (80 % de las imágenes con displasia y 30 % de las imágenes sin displasia) y Test (20 % de las imágenes con displasia y las imágenes sin displasias más próximas a las displasias) Método Próximas.

de espacio, se insertan en las Figuras 3.4, 3.4 y 3.4 la matriz de imágenes Flair de cada método, las matrices de tipo T1 serán muy similares a estas y se podrá ver en el código entregado.

- Matriz de imágenes Flair del Método aleatorio se observan en la Figura 3.4
- Matriz de imágenes Flair del Método Leave One Out se observan en la Figura 3.4
- Matriz de imágenes Flair del Método próximas se observan en la Figura 3.4

La elección para este proyecto ha sido emplear la red EfficientNet B0, ya que como hemos comentado en el capítulo anterior es la mejor red en cuanto a eficacia y precisión cuando se tiene un número de imágenes tan reducido como es nuestro caso. La creación del modelo será el mismo para todos los métodos de selección de imágenes que emplearemos como se puede apreciar en la Figura 25.

Se puede observar que los pesos serán los de imagenet, se eliminará la última capa para poder realizar el transfer learning. De esta manera nuestro modelo será capaz de reconocer los conjuntos de displasia y no displasia, por último, indicamos cual será la forma que tendrán las imágenes de entrada.

En el siguiente paso se creará la última capa que nos indicará por medio de su salida si las imágenes introducidas poseen o no la anomalía y se unirá a la red creada anteriormente.

Para esta última capa se empleará el Global Average Pooling 2D explicado previamente como subsampling, además, aplicaremos un método llamado Dropout que desactiva un numero de neuronas de una red neuronal de forma aleatoria, como nuestra intención es mantener la mayoría de neuronas activadas y en capas ocultas usaremos el valor de 0.5. Por otro lado, usaremos la función Dense que nos añadirá esta última capa de 2 neuronas (ya que nuestro clasificador detecta si hay o no hay displasia) y su función de activación será softmax ya explicada previamente. Por último, haremos que el modelo tenga por entrada la entrada de la red y como salida esta última capa creada. Esta parte del código está regida en las Figuras 26 y 27

Para la evaluación de los modelos se calculará el valor de pérdida de la entropía cruzada entre las etiquetas y las predicciones, además, usaremos como métrica los valores de precisión obtenidos en las predicciones de cada entrenamiento para determinar cual será el mejor de todos los modelos.

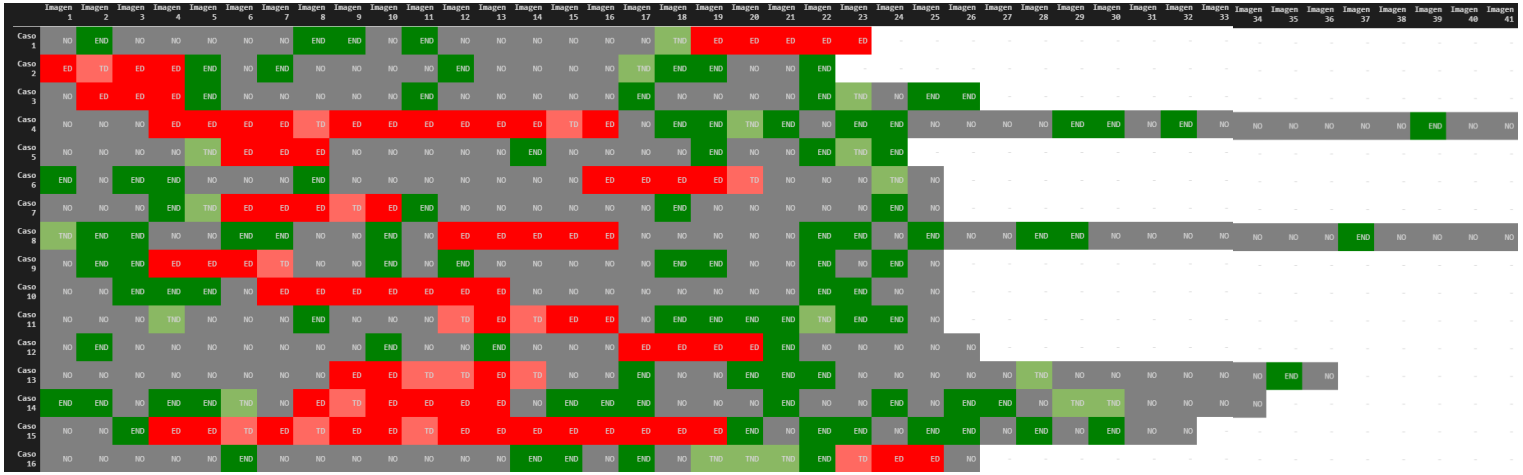

Figura 22: Conjunto Flair Método Aleatorio. Entrenamiento Displasia (ED): Color Rojo. Entrenamiento No Displasia (END): Color Verde. Test Displasia (TD): Color Rojo Claro. Test No Displasia (TND): Color Verde Claro. Imagen no usada (NO): Color gris. Cuando el paciente no tiene más imágenes (-): Color blanco.

Figura 23: Conjunto Flair Método Leave One Out Caso 10. Entrenamiento Displasia (ED): Color Rojo. Entrenamiento No Displasia (END): Color Verde. Test Displasia (TD): Color Rojo Claro. Test No Displasia (TND): Color Verde Claro. Imagen no usada (NO): Color gris. Cuando el paciente no tiene más imágenes (-): Color blanco.

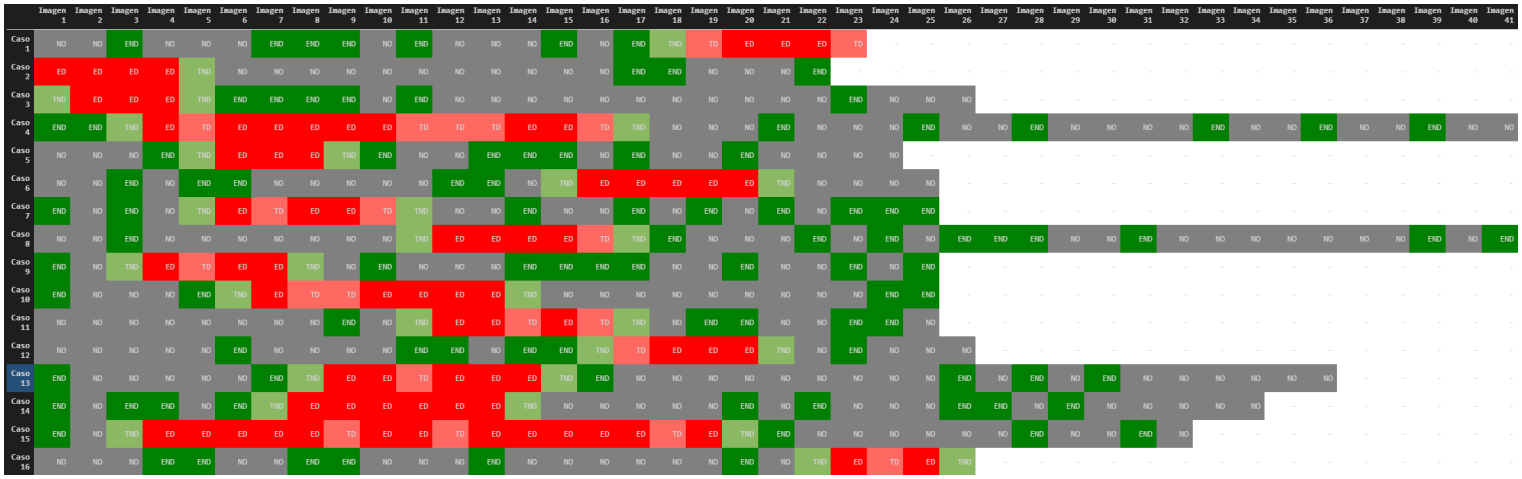

Figura 24: Conjunto Flair Método Próximas. Entrenamiento Displasia (ED): Color Rojo. Entrenamiento No Displasia (END): Color Verde. Test Displasia (TD): Color Rojo Claro. Test No Displasia (TND): Color Verde Claro. Imagen no usada (NO): Color gris. Cuando el paciente no tiene más imágenes (-): Color blanco.

```
effnet = EfficientNetB0(weights = "imagenet",include_top=False,
    input_shape=(img_size,img_size,3))
```

Figura 25: Creación de la red EfficientNet B0.

```
model = effnet.output
model = GlobalAveragePooling2D()(model)
model = Dropout(0.5)(model)
model = Dense(2,activation = "softmax")(model)
model = Model(inputs = effnet.input,outputs = model)
model.summary()
```

Figura 26: Código de la construcción de la última capa y creación del modelo.

Utilizaremos un método para reducir el learning rate en caso de que cuando pasen 2 épocas sin mejorar la precisión se disminuya este valor y así comprobar si puede mejorar el entrenamiento de la máquina. Se puede ver en la Figura 28 cómo se creó esta parte del código.

Una vez realizado el paso de selección de imágenes, creado nuestro modelo y haber establecido el modelo de compilación, ya estamos en situación de realizar los entrenamientos y así descubrir cuál de todos los modelos será el mejor y el que por tanto usaremos finalmente en nuestro detector.

Para el entrenamiento emplearemos el conjunto de Entrenamiento con las imágenes que poseen y que no poseen displasia, el número de épocas que usaremos será 15 y utilizaremos como datos de validación el conjunto de test. Es cierto que cuando se habla de datos de validación se suele usar un subconjunto del conjunto de entrenamiento, pero según la información que se ha podido obtener de Keras, este conjunto de validación (*validation_data*) no se emplea para entrenar, sino como un método de evaluación para saber si se tiene que disminuir el learning rate. Para estos entrenamientos se tendrán en cuenta el checkpoint y la reducción del learning rate comentados previamente. En las Figuras 29 y 30 podemos apreciar el proceso de entrenamiento del modelo.

Repetimos este proceso de entrenamiento 10 veces para cada método de selección de imágenes y comparamos los resultados obtenidos. A continuación se adjuntan unas tablas con los valores medios obtenidos de de cada uno de los métodos de selección de imágenes realizados.

Estos valores que se podrán observar serán:

- Pérdida: Es un número que indica cuánto de incorrecto ha sido la predicción del modelo

Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	[(None, 224, 224, 3)	0	
rescaling_8 (Rescaling)	(None, 224, 224, 3)	0	input_9[0][0]
normalization_8 (Normalization)	(None, 224, 224, 3)	7	rescaling_8[0][0]
stem_conv_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	normalization_8[0][0]
stem_conv (Conv2D)	(None, 112, 112, 32)	864	stem_conv_pad[0][0]
stem_bn (BatchNormalization)	(None, 112, 112, 32)	128	stem_conv[0][0]
stem_activation (Activation)	(None, 112, 112, 32)	0	stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D	(None, 112, 112, 32)	288	stem_activation[0][0]
block1a_bn (BatchNormalization)	(None, 112, 112, 32)	128	block1a_dwconv[0][0]

Figura 27: Primeras capas de nuestro modelo.

```
model.compile(optimizer=Adam(lr=0.0001),loss="categorical_crossentropy",metrics = ["accuracy"])
checkpoint = ModelCheckpoint("effnetPROXIMAS10.h5",monitor="val_accuracy",save_best_only=True,mode="auto",verbose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3,
                              patience = 2, min_delta = 0.001,
                              mode = 'auto', verbose = 1)
```

Figura 28: Código del Modelo de compilación.

en un solo ejemplo.

- Precisión: Se refiere a lo cerca que está el resultado de una medición del valor verdadero. Se representa como la proporción de resultados verdaderos dividido entre el número total de casos examinados .

- Verdaderos Positivos: Probabilidad con la que a una imagen con displasia se le clasificaría como displasia.

- Falsos Positivos: Probabilidad con la que a una imagen sin displasia se le clasificaría como displasia.

- Verdaderos Negativos: Probabilidad con la que a una imagen sin displasia se le clasifi-

```
history = model.fit(X_train,y_train,epochs=15,validation_data = (X_test,y_test),verbose=1,
    callbacks=[checkpoint,reduce_lr])
```

Figura 29: Código del entrenamiento.

```
Epoch 1/15
18/18 [==============================] - 152s 8s/step - loss: 0.6975 - accuracy: 0.6100 - val_loss: 0.7113 - val_accuracy: 0.5221

Epoch 00001: val_accuracy improved from -inf to 0.52212, saving model to effnetPROXIMAS10.h5
Epoch 2/15
18/18 [==============================] - 143s 8s/step - loss: 0.4669 - accuracy: 0.8038 - val_loss: 0.7035 - val_accuracy: 0.5398

Epoch 00002: val_accuracy improved from 0.52212 to 0.53982, saving model to effnetPROXIMAS10.h5
Epoch 3/15
18/18 [==============================] - 142s 8s/step - loss: 0.3838 - accuracy: 0.8528 - val_loss: 0.7010 - val_accuracy: 0.5841

Epoch 00003: val_accuracy improved from 0.53982 to 0.58407, saving model to effnetPROXIMAS10.h5
Epoch 4/15
18/18 [==============================] - 141s 8s/step - loss: 0.3281 - accuracy: 0.8615 - val_loss: 0.7117 - val_accuracy: 0.5575
```

Figura 30: Primeras épocas del entrenamiento del conjunto 10 del Método Próximas

caría como no displasia.

- Falsos Negativos: Probabilidad con la que a una imagen con displasia se le clasificaría como no displasia.

Método	Pérdida	Precisión	+/+	+/-	-/-	-/+
Aleatorio	0.5398±0,0589	0.7287±0,0495	0.712±0,161	0.288±0,161	0.744±0,143	0.256±0,143
Leave One Out	0,8473±0,1814	0,5308±0,1119	0.361±0,346	0.639 ±0,346	0.707±0,273	0.293±0,273
Próximas	0.7631±0,0662	0.5575±0,0273	0.729±0,154	0.271±0,154	0.400±0,136	0.600±0,136

3.5. Evaluación

En esta etapa del proyecto, se han construido uno o varios modelos que parecen alcanzar la calidad suficiente desde una perspectiva de análisis de datos, serán evaluados por los expertos para ver si se alcanzan los objetivos de la primera etapa.

Se realizó el día 21 de junio de 2021 una reunión con las doctoras Beatriz Asenjo y Victoria Fernández para mostrarles el resultado obtenido a lo largo de estos meses de estudio. Esta reunión nos sirvió para además de ver qué les parecía el trabajo realizado, comentar en conjunto qué aspectos cambiar o añadir para mejorar nuestro modelo en un futuro.

De la reunión se pudo sacar en claro varios aspectos entre los que cabe destacar:

- Consideraban un buen resultado como primera iteración de ciclo CRISP-DM que la precisión del modelo final sobre el último paciente facilitado fuera del 84.46 %, ya que por lo que pudimos saber, se habían realizado detectores previamente sobre otro tipo de enfermedades como podía ser el cáncer de mama que tan solo contaban con un 24 % de precisión. En este sentido nos aclararon que el modelo obtenido resultaría de gran ayuda en un caso real. Al no ser del todo preciso este modelo, las doctoras señalaron que lo consideraban como una buena herramienta para hacer que el experto se centre en aquel grupo de imágenes continuas que nuestro detector predice como displasia y así dictaminar si el paciente posee o no la anomalía, ayudándole a no tener que revisar con la misma profundidad aquellas imágenes que nuestro modelo pueda llegar a señalar con displasia siendo casos aislados dentro del conjunto de imágenes.

- Las doctoras comentaron que si en un futuro se continúa con el proyecto se asegurarían de indicarnos qué imágenes tanto Flair como T1 corresponden a la misma porción del cerebro, porque si en una determinada imagen Flair hay una displasia, en la correspondiente de T1 debería haberla también, esto podría darnos pistas para cómo mejorar nuestro entrenamiento.

- Uno de los aspectos que se pudieron dar por sentados es que en el mundo de la medicina es peor que haya mayor número de falsos negativos que falsos positivos, esto nos servirá en un futuro para centrarnos en cómo deberíamos entrenar futuros modelos para que estos resultados se vean disminuidos. La explicación de esto es que si se obtiene algún falso negativo, este tipo de sistema haría despertar la atención del experto y este se vería obligado a tenerlo en cuenta, mientras que si es un falso negativo, esta imagen podría llegar a pasar desapercibida por el experto.

- Un tema que consideraban que se podría tener en cuenta para un futuro es separar nuestro detector en un detector de imágenes Flair y en otro detector de imágenes T1. Las doctoras nos explicaron que las imágenes Flair poseen una calidad mucho mayor que las imágenes en T1, por tanto, podemos llegar a pensar que si se realiza una separación de ambos tipos de imágenes la precisión podría llegar a aumentar.

- Otro aspecto para mejorar los resultados obtenidos en este modelo es que se acordó man-

dar más imágenes en el caso de que este proyecto siguiese adelante tras la realización de este Trabajo Fin de Grado.

- En cuanto a la parte más visual del sistema que es la interfaz, la consideraban correcta para el tipo de uso que tiene. Como apunte nos señalaron que una buena idea sería la de añadir una funcionalidad de que haciendo clic al nombre de una determinada imagen, nos abra una ventana con la imagen en cuestión por si el experto quiere realizar algún tipo de comparación o detectar si existe displasia por si mismo en caso de duda. Por último, otro aspecto que se comentó que se podría añadir es que además de la predicción de si la imagen posee la anomalía o no, que el sistema sea capaz de indicar en qué porcentaje está seguro nuestro modelo que la imagen posee displasia o no, esto es algo que se tendrá en cuenta para las Líneas futuras ya que lo consideramos muy útil de cara al experto.

3.6. Uso

El conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo.

Como se puede observar en las tablas anteriores los resultados obtenidos son algo bajos debido, entre otras causas, al bajo número de imágenes disponibles para el entrenamiento. En el apartado de Lineas Futuras dentro de las conclusiones se comentan en mayor profundidad los siguientes pasos que se podrían dar para alcanzar mejores resultados en futuras iteraciones del proceso CRISP-DM.

De los tres métodos de selección de imágenes podemos observar que los mejores resultados se encuentran en el Método Aleatorio con una media de precisión del 72.87 %. Por tanto, para el modelo definitivo usaremos como método de selección de imágenes sin displasia el Método Aleatorio, mientras que para el conjunto de entrenamiento de displasia se usarán todas las imágenes facilitadas por las doctoras en las que aparece la displasia . Como el conjunto de entrenamiento de displasia cuenta ahora con 272 imágenes, el conjunto de entrenamiento de no displasia aumentará a los anteriores modelos también, siendo el 40 % de las imágenes totales sin displasia que nos han podido dar las doctoras.

Este modelo definitivo será el que se empleará de manera interna para realizar las pre-

dicciones de los nuevos pacientes en los que se necesiten detectar la displasia, todo esto será integrado en una interfaz que se presentará en el manual de usuario para que los expertos puedan emplear el sistema sin necesidad alguna de tener conocimientos informáticos.

En último momento las doctoras nos han podido facilitar otro caso, el número 17, que constaba de 103 imágenes de las cuales 11 poseían displasia. Hemos realizado la predicción del paciente y el resultado ha sido el que se muestra en la siguiente tabla.

Precisión	+/+	+/-	-/-	-/+
0,8446	0,6363	0,3636	0,8695	0,1304

4

Conclusiones y Líneas Futuras

4.1. Conclusiones

Para comenzar este apartado se debe destacar que uno de los principales problemas que se han encontrado en este trabajo ha sido el escaso número de imágenes que poseíamos para realizar los entrenamientos. En un proyecto de minería de datos de este tipo se necesita contar con un banco de imágenes mucho mayor que las 272 imágenes con displasia y 1161 imágenes sin displasia que nos han podido facilitar. Un ejemplo de ello es un clasificador de tumores cerebrales que se encuentra en Kaggle que cuenta con más de 800 imágenes en cada uno de los tumores cerebrales que clasificaba [2].

Debido a este número de imágenes nos vimos forzados a buscar una red cuya característica principal fuese tener una gran precisión con un bajo número de datos y por eso usamos desde un primer momento la red EfficientNet, cuando cabe la posibilidad de que existiese alguna red más común para este tipo de trabajos con imágenes de resonancias magnéticas.

Estos son los principales motivos por los que pensamos que los resultados obtenidos de los modelos no son tan buenos como se podía llegar a esperar desde un primer momento, pero al fin y al cabo consideramos que obtener un valor de media de precisión del 72.87 % en el método aleatorio de selección de imágenes es un magnífico resultado a pesar de las circunstancias en las que se ha desarrollado el proyecto.

El valor de la precisión del modelo final con el conjunto definitivo de imágenes de entrenamiento deberá ser mayor que los resultados obtenidos previamente, esto se ve reflejado en que la precisión de la predicción realizada sobre el último caso que las doctoras nos pudo facilitar fue de un 84.466 %, mejorando todos los modelos anteriores obtenidos.

Cuando se les presentaron los resultados a las doctoras, estas estaban bastante contentas como resultado de esta primera iteración del ciclo CRISP-DM y se ha mostrado interés en continuar con este proyecto tras la finalización del TFG para llegar en un futuro a que nuestro sistema se convierta en una herramienta real para la detección de este tipo de anomalías.

4.2. Líneas Futuras

Como líneas futuras se ha pensado sobre la manera de poder mejorar los entrenamientos y la forma de perfeccionar los modelos:

- Obtener un mayor número de imágenes con las que poder realizar el entrenamiento.
- Un mejor preprocesamiento de la imagen, se ha pensado que podría mejorar ayudar a mejorar la detección de la displasia que en las imágenes solo apareciera la parte cerebral de la resonancia y eliminar todo lo demás.
- Emplear herramientas como labelImage que nos permitirían etiquetar elementos dentro de una imagen, así los especialistas podrían señalar dónde se localiza la displasia explícitamente y modificar nuestra manera de plantear el estudio.
- Encontrar las correspondencias de la misma porción de cerebro entre las imágenes Flair y T1 para encontrar pistas sobre cómo mejorar nuestro entrenamiento.
- Conseguir un modelo en el que se vean reducido el número de falsos positivos sobre el número de falsos negativos debido al coste médico entre ambos grupos.
- Indagar en qué otro tipo de redes más específicas en la detección de elementos en imágenes médicas puede emplearse para este trabajo.
- Realizar un clasificador para imágenes Flair y otro para imágenes T1 y comprobar si de esta manera se obtienen mejores resultados que en un clasificador con ambos tipos de imágenes.
- Mejorar la interfaz del usuario, añadiéndole funcionalidades y perfeccionándola visualmente. Por petición de las doctoras se le piensa incluir un porcentaje de predicción de cada imagen y la posibilidad de ver la imagen directamente desde la interfaz sin necesidad de tener que navegar entre las carpetas.

Referencias

[1] *2D Average pooling block | Peltarion Platform.* URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-average-pooling (visitado 07-05-2021).

[2] *brain tumor classification v3.2 | Kaggle.* URL: https://www.kaggle.com/astaroth88/brain-tumor-classification-v3-2 (visitado 15-06-2021).

[3] *Conceptos básicos de ayuda de CRISP-DM - Documentación de IBM.* URL: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=dm-crisp-help-overview (visitado 25-05-2021).

[4] *Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning.* URL: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/ (visitado 04-05-2021).

[5] *EfficientNet: Repensar el escalado del modelo para redes neuronales convolucionales (clasificación de imágenes).* URL: https://ichi.pro/es/efficientnet-repensar-el-escalado-del-modelo-para-redes-neuronales-convolucionales-clasificacion-de-imagenes-263228629273165 (visitado 19-05-2021).

[6] *Función de activación ReLU. | Download Scientific Diagram.* URL: https://www.researchgate.net/figure/Figura-18-Funcion-de-activacion-ReLU_fig5_328600321 (visitado 14-05-2021).

[7] *Google AI Blog: EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling.* URL: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html (visitado 19-05-2021).

[8] *Intro a las redes neuronales convolucionales | by Bootcamp AI | Medium.* URL: https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8 (visitado 04-05-2021).

[9] Lohith G. Kini, James C. Gee y Brian Litt. “Computational analysis in epilepsy neuroimaging: A survey of features and methods”. En: *NeuroImage: Clinical* 11 (2016), págs. 515-529. ISSN: 22131582. DOI: 10.1016/j.nicl.2016.02.013. URL: http://dx.doi.org/10.1016/j.nicl.2016.02.013.

Apéndice A
Manual de Usuario

En este anexo se explica cómo usar la aplicación del detector de displasias. En la Figura 31 se presenta la interfaz que utilizará el usuario para predecir una observación nueva o para cambiar el modelo actual por uno de mayor calidad.

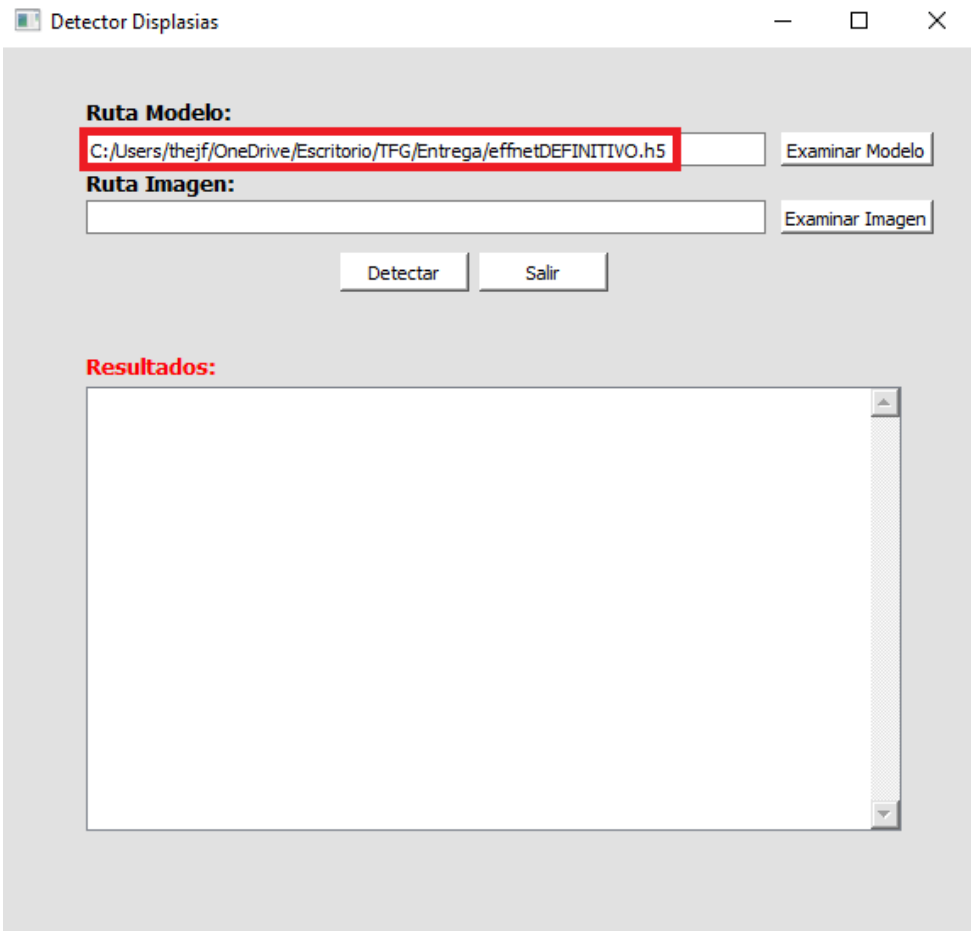

Figura 31: Interfaz del detector de displasias

Como vimos en el apartado Aplicación de CRISP-DM 3.6 el modelo final generado con todas las imágenes de displasias como entrenamiento será usado por defecto para realizar las predicciones de los nuevos pacientes de los que sea necesario detectar la displasia. Si se quiere seleccionar otro modelo, se hace clic en *Examinar Modelo*, se nos abrirá una ventana con los archivos que se encuentran en el equipo donde se ejecuta la aplicación y se cambiará por el

modelo que seleccionemos.(Figura 32)

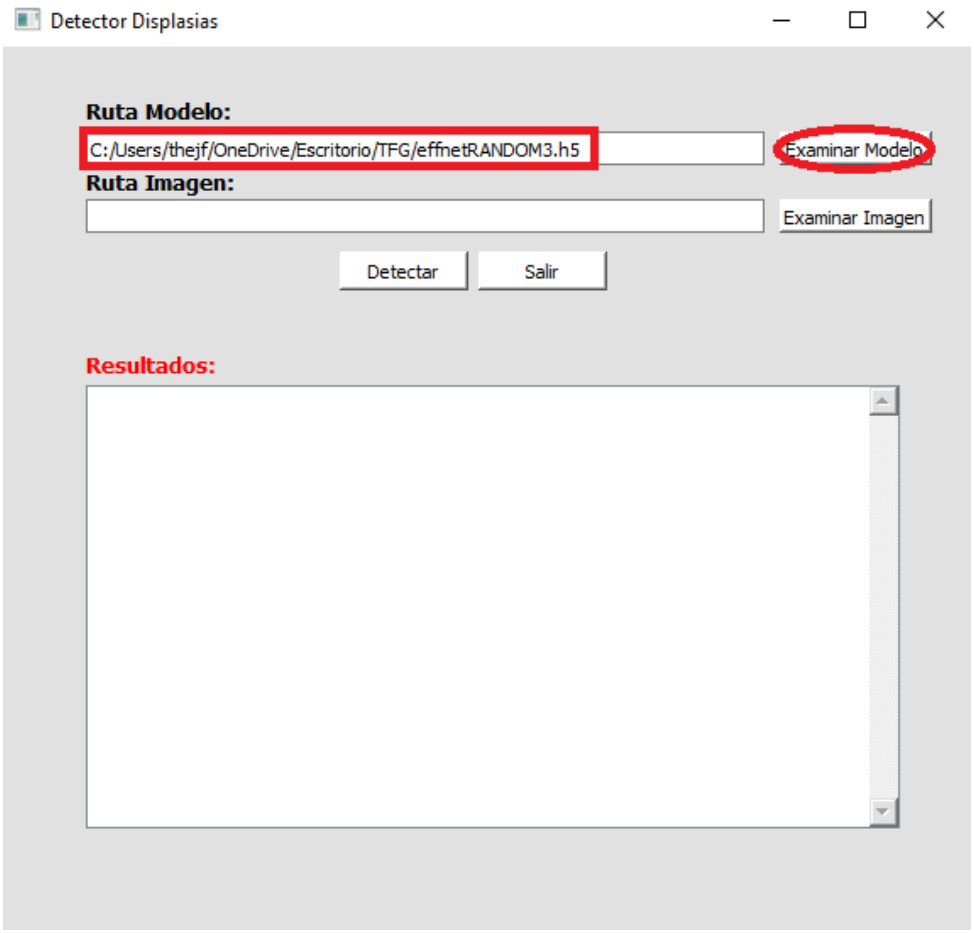

Figura 32: Cambio de modelo

El siguiente paso que se debe realizar es seleccionar una carpeta en formato zip con las imágenes jpg del nuevo paciente o bien seleccionar aquella imagen en formato jpg que queramos comprobar si aparece o no la displasia. Para ello hay que clicar en *Examinar Imagen*, se nos abrirá una ventana con los archivos que se encuentran en el equipo donde se ejecuta la aplicación y se cogerá la ruta donde se encuentra el archivo zip. (Figura 33)

Una vez que ya se han realizado los pasos anteriores, ya estamos en disposición de pulsar en el botón *Detectar* y a los pocos segundos se nos cargará en la parte de Resultados una tabla con las imágenes del paciente introducido y las predicciones de nuestro modelo para dichas imágenes. (Figura 34)

Si en algún momento se desea salir de la aplicación se le podrá dar a la cruz de la ventana o bien al botón *Salir*.

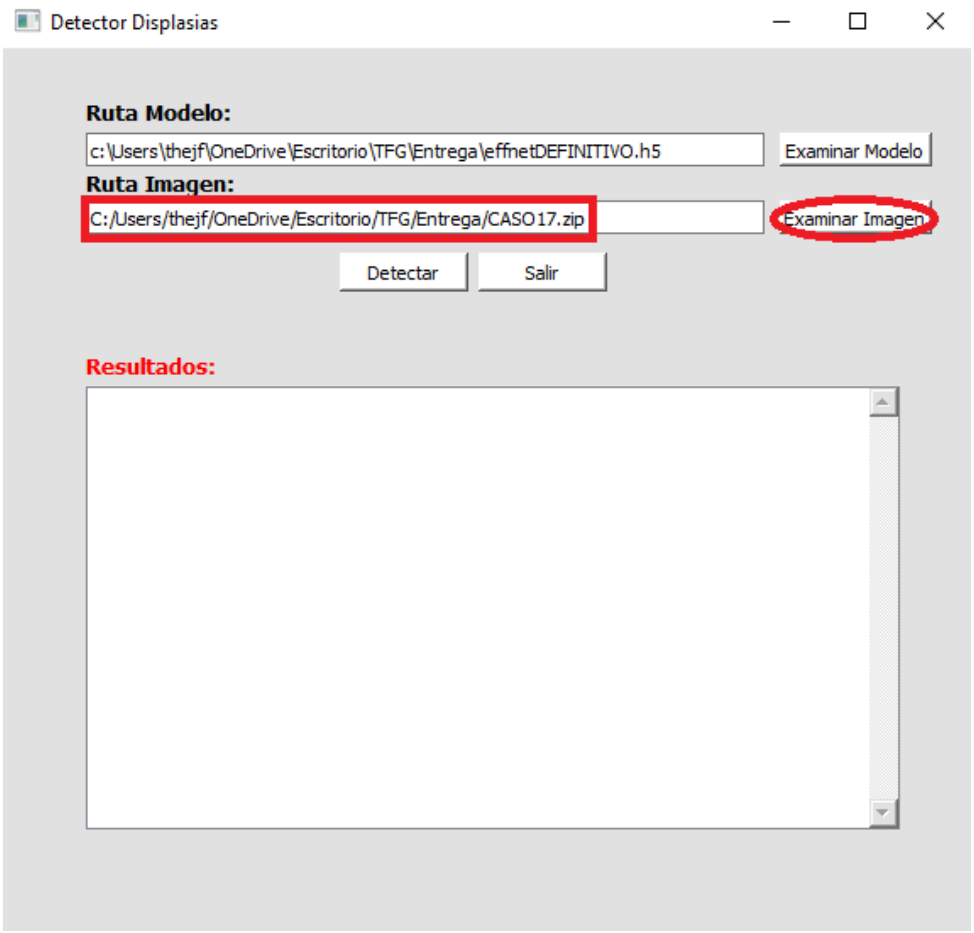

Figura 33: Elección de la Ruta de la Imagen

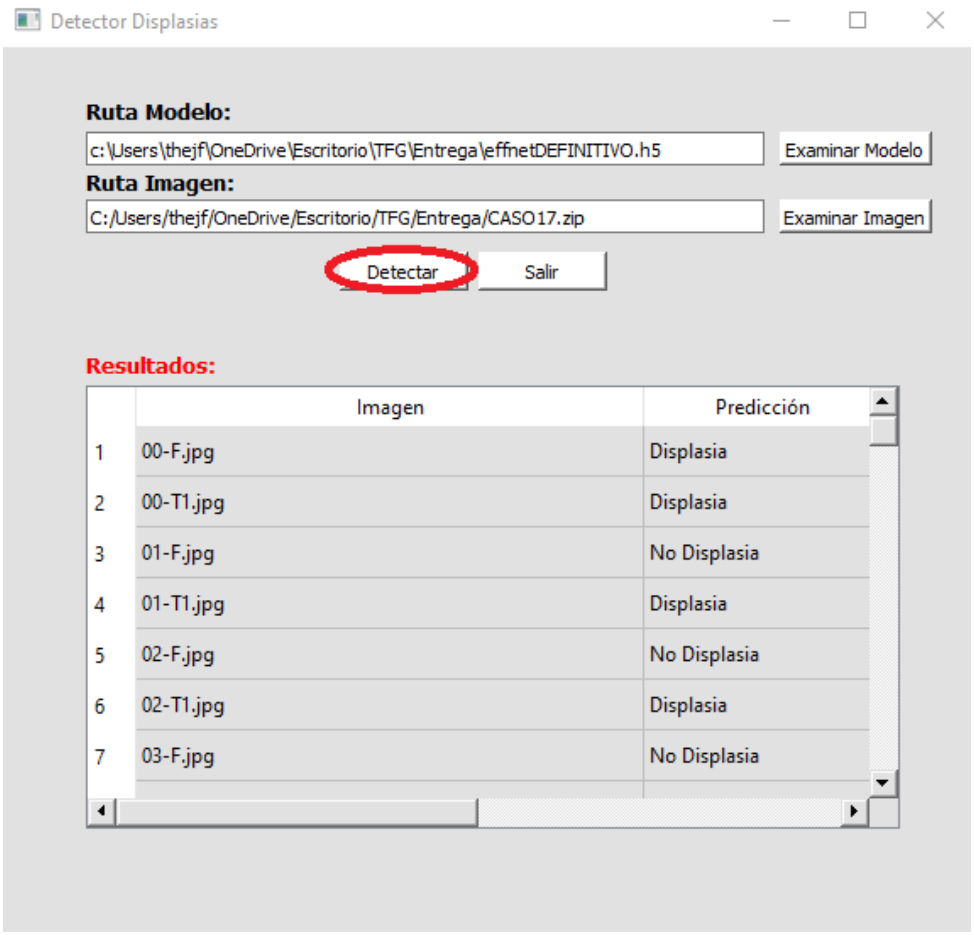

Figura 34: Resultados de la predicción

Apéndice B
Entrenamientos realizados

En este apartado se mostrará los resultados obtenidos por los entrenamientos a lo largo de este proyecto.

- **Método Aleatorio**

Pérdida	Precisión	+/+	+/-	-/-	-/+
0.5251	0.6759	0.85	0.15	0.50	0.50
0.6358	0.6481	0.41	0.59	0.89	0.11
0.4449	0.7962	0.81	0.19	0.78	0.22
0.5404	0.7222	0.78	0.22	0.67	0.33
0.5802	0.6944	0.80	0.20	0.59	0.41
0.6240	0.7037	0.52	0.48	0.89	0.11
0.5245	0.7777	0.72	0.28	0.83	0.17
0.5291	0.7222	0.59	0.41	0.85	0.15
0.5019	0.7777	0.70	0.30	0.85	0.15
0.4916	0.7685	0.94	0.06	0.59	0.41

- **Método Leave One Out**

Pérdida	Precisión	+/+	+/-	-/-	-/+
0.8017	0.3913	0.45	0.55	0.33	0.67
0.7988	0.4888	0.05	0.95	0.91	0.09
0.6097	0.7692	0.83	0.17	0.71	0.29
0.6834	0.6075	0.31	0.69	0.90	0.10
0.8163	0.4799	0.75	0.25	0.23	0.77
1.1742	0.4545	0.00	1.00	0.88	0.12
1.1134	0.5217	0.00	1.00	1.00	0.00
0.7006	0.6470	0.88	0.12	0.44	0.56
0.8372	0.4864	0.17	0.83	0.79	0.21
0.9379	0.4615	0.17	0.83	0.88	0.12

- **Método de las más próximas**

Pérdida	Precisión	+/+	+/-	-/-	-/+
0.7537	0.5752	0.74	0.26	0.42	0.58
0.6951	0.5398	0.56	0.44	0.53	0.47
0.7937	0.5663	0.91	0.09	0.25	0.75
0.7336	0.5752	0.69	0.31	0.47	0.53
0.9088	0.5752	0.80	0.20	0.37	0.63
0.7377	0.5132	0.54	0.46	0.49	0.51
0.7156	0.6017	0.87	0.13	0.36	0.64
0.8139	0.5309	0.83	0.17	0.25	0.75
0.6888	0.5663	0.48	0.52	0.64	0.36
0.7901	0.5309	0.87	0.13	0.22	0.78

Apéndice C
Cómo entrenar nuevos modelos

La metodología realizada para este proyecto ha sido CRISP-DM, una metodología que permite la realización de más de una iteración, pudiendo así mejorar el sistema con los posibles modelos generados en un futuro. Además, nuestra interfaz permite la selección del modelo con el que realizar las predicciones, lo que favorece a que si en una futura iteración se encuentra un modelo mejor que el actual, se pueda realizar la predicción con este nuevo modelo sin ningún tipo de problema.

Este anexo explica los pasos necesarios para generar un modelo y así poder introducirlo en nuestro sistema.

Antes de nada es necesario señalar cuáles serán las librerías necesarias para la creación del modelo, muchas de ellas son de tratamiento de imágenes, pero entre todas las librerías cabe destacar keras que es que la principal responsable de la creación del modelo. Estas librerías están reflejadas en la Figura 35.

Como nuestro conjunto de entrenamiento final de displasia estará formado por todas aquellas imágenes que nos han facilitado las doctoras que poseían la anomalía, debemos centrarnos en cómo obtener el conjunto de imágenes de entrenamiento final sin displasia. Debido a que el método de selección de imágenes con mejor valor medio en los entrenamientos realizados ha sido el método aleatorio, hemos decidido que este sea el método de selección para nuestro modelo final. En la Figura 36 podemos observar cómo se ha realizado la selección.

En un primer momento esta parte del código lo que hace es contar cuántas imágenes sin displasia poseemos. El siguiente paso consiste en realizar un bucle en el que vamos a ir generando números aleatorios, esos números serán las posiciones de las imágenes sin displasia dentro de la carpeta que contiene este tipo de imágenes. Además en cada iteración del bucle si no nos pasamos del número de imágenes necesarias para el entrenamiento (en este caso un 40 % del total de imágenes sin displasia ya que como recordamos hay un gran desbalance entre

```
import os
import random
from tqdm import tqdm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import cv2
from tensorflow.keras.preprocessing.image import load_img,ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing import image

from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.applications import EfficientNetB0

from tensorflow.keras import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten,Dense,Conv2D,MaxPooling2D,Dropout,GlobalAveragePooling2D

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping,ReduceLROnPlateau

import glob
import shutil
```

Figura 35: Librerías empleadas en el proyecto.

```
nimagenesND=len(glob.glob("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\NoDisplasia\\*.jpg"))
cont=0
for i in range (0, nimagenesND):
    n = random.randint(0, nimagenesND-1)
    if cont <= nimagenesND*0.4 and not (os.path.isfile("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\Clasificador\\Entrenamiento\\No
Displasia\\" + os.listdir("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\NoDisplasia\\")[n])):
        shutil.copy("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\NoDisplasia\\" + os.listdir
("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\NoDisplasia\\")[n],"C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\Clasificador\\Entrenamiento\\No
Displasia\\" + os.listdir("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\NoDisplasia\\")[n])
        cont += 1
```

Figura 36: Selección de imágenes sin displasia para el conjunto de entrenamiento final.

el número de imagenes con y sin displasia) y si la imagen no está repetida dentro del conjunto de entrenamiento, se introducirá dentro de nuestro conjunto Entrenamiento No Displasia.

Una vez que tengamos el conjunto de entrenamiento formado nos adentramos en la creación del modelo y del entrenamiento.

Guardamos en variables las rutas necesarias tanto del clasificador como dónde se encuentran las imágenes de entrenamiento, esto se puede ver en la Figura 37.

```
root = "C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\Clasificador"
classes_ = os.listdir("C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\Clasificador\\Entrenamiento")
train_dir = "C:\\Users\\thejf\\OneDrive\\Escritorio\\TFG\\Clasificador\\Entrenamiento\\"
```

Figura 37: Rutas necesarias.

El siguiente paso es cargar las imágenes de entrenamiento como se muestra en la Figura 38. En ella podemos ver que se crean dos conjuntos uno que va a ser X_Train y otro que se

va a llamar y_train, en el primer conjunto se van a almacenar las imágenes y en el segundo la etiqueta de cada imagen, es decir, si posee o no posee displasia.

```
files_path_dict = {}
for class_ in os.listdir(train_dir):
    files_path_dict[class_] = list(map(lambda x: train_dir+class_+'/'+x, os.listdir(train_dir+class_)))
```

```
train_Displasia = train_dir + "Displasia"
train_NoDisplasia = train_dir + "No Displasia"

X_train = []
y_train = []

for i in tqdm(os.listdir(train_Displasia)):
    path = os.path.join(train_Displasia,i)
    img = cv2.imread(path)
    img = cv2.resize(img,(224,224))
    X_train.append(img)
    y_train.append('Displasia')

for i in tqdm(os.listdir(train_NoDisplasia)):
    path = os.path.join(train_NoDisplasia,i)
    img = cv2.imread(path)
    img = cv2.resize(img,(224,224))
    X_train.append(img)
    y_train.append('No Displasia')
```

```
100%|██████████| 272/272 [00:01<00:00, 138.02it/s]
100%|██████████| 465/465 [00:03<00:00, 142.92it/s]
```

Figura 38: Carga de las imágenes de entrenamiento.

Para ver qué se ha cargado realmente, se ha realizado el código mostrado en la Figura 39 que nos permite saber cuántas han sido las imágenes cargadas de entrenamiento.

A continuación se realiza una codificación para poder realizar el entrenamiento y las predicciones, ya que el ordenador no trabaja con nombres sino con números. El valor 0 serán aquellas imágenes con Displasia y el valor 1 serán las imágenes No Displasia, para que sea

```
X_train = np.array(X_train) #
y_train = np.array(y_train)

print("Image shape:", X_train.shape[1:3])
print("Number of Training samples:", X_train.shape[0])
```

```
Image shape: (224, 224)
Number of Training samples: 737
```

```
plt.style.use("ggplot")
plt.figure(figsize=(9,5))
sns.countplot(x=y_train)
plt.title("Train class balance")
plt.show()
```

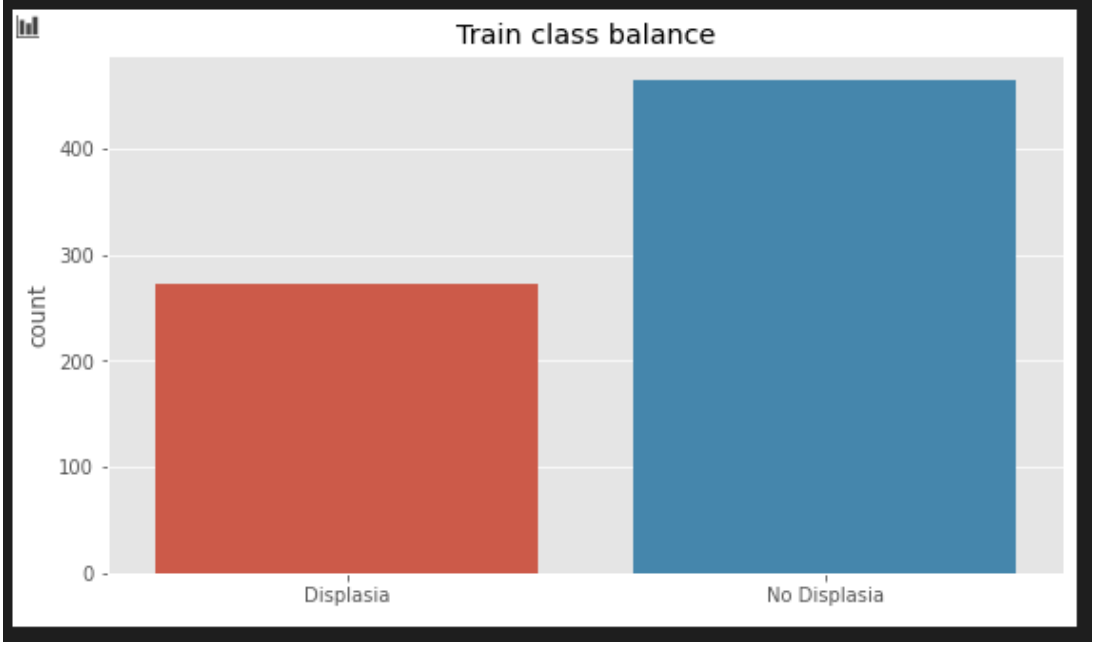

Figura 39: Imágenes cargadas de entrenamiento.

más visual cuando imprimimos transformamos los números a Displasia y No Displasia. Todo esto se puede observar en la Figura 40.

```
LE = LabelEncoder()
y_train = LE.fit_transform(y_train)
```

```
print("Nº de elementos en y_train:", y_train.shape[0])
print("Unique Counts:")
unique,counts = np.unique(LE.inverse_transform(y_train),return_counts=True)
print(unique,counts)
```

```
y_train = to_categorical(y_train)
y_train.shape
```

Figura 40: Proceso de codificación de las imágenes.

Como estamos hablando del modelo final usado no poseemos conjunto de test como tal, por tanto lo que se suele usar como test en este caso son las imágenes del conjunto de entrenamiento con cambios en sus dimensiones, rotación o zoom entre otros factores para obtener el conjunto de test. Las transformaciones realizadas para este caso estará recogido en la Figura 41.

Una vez que tenemos ya los dos conjuntos tanto de entrenamiento como de test, ahora se tiene que continuar con la creación de nuestra red y modelo.

A partir de aquí se pueden realizar numerosos cambios para ver qué red o modelo es el mejor, en este caso se ha partido de una red EfficientNet B0, la última capa que hemos añadido a esta red usa Global Average Pooling 2D y una función de activación softmax. Esto se puede observar en la Figura 42.

Una vez creada la red ya se puede pasar a la compilación del modelo, ahí introduciremos cuál será nuestro optimizador, en este caso Adam con un learning rate de 0.0001 y qué parámetros usaremos de métrica, en este proyecto será la precisión y la pérdida, esta última sigue una función de entropía cruzada. Se crearán los parámetros checkpoint que nos indica dónde se guardará nuestro modelo (siempre en un archivo .h5) y reduce_lr que nos indicará en cuánto se verá reducido el learning rate cuando en un número de épocas (también introducido por

```
img_size=224
train_datagen = ImageDataGenerator(rotation_range=30,height_shift_range=0.2,
                                   zoom_range = 0.3,horizontal_flip=True)

train_gen = train_datagen.flow_from_directory(directory = train_dir,target_size=(img_size,img_size),class_mode ="categorical",
                                              batch_size=32)
```

```
Found 737 images belonging to 2 classes.
```

```
sample_x,sample_y = next(train_gen)

plt.figure(figsize=(12,9))
for i in range(6):
    plt.subplot(2,3,i+1)
    sample = image.array_to_img(sample_x[i])
    if np.argmax(sample_y[i]) == 0 :
        plt.xlabel("Displasia")
    else:
        plt.xlabel("No Displasia")
    plt.imshow(sample)
plt.show()
```

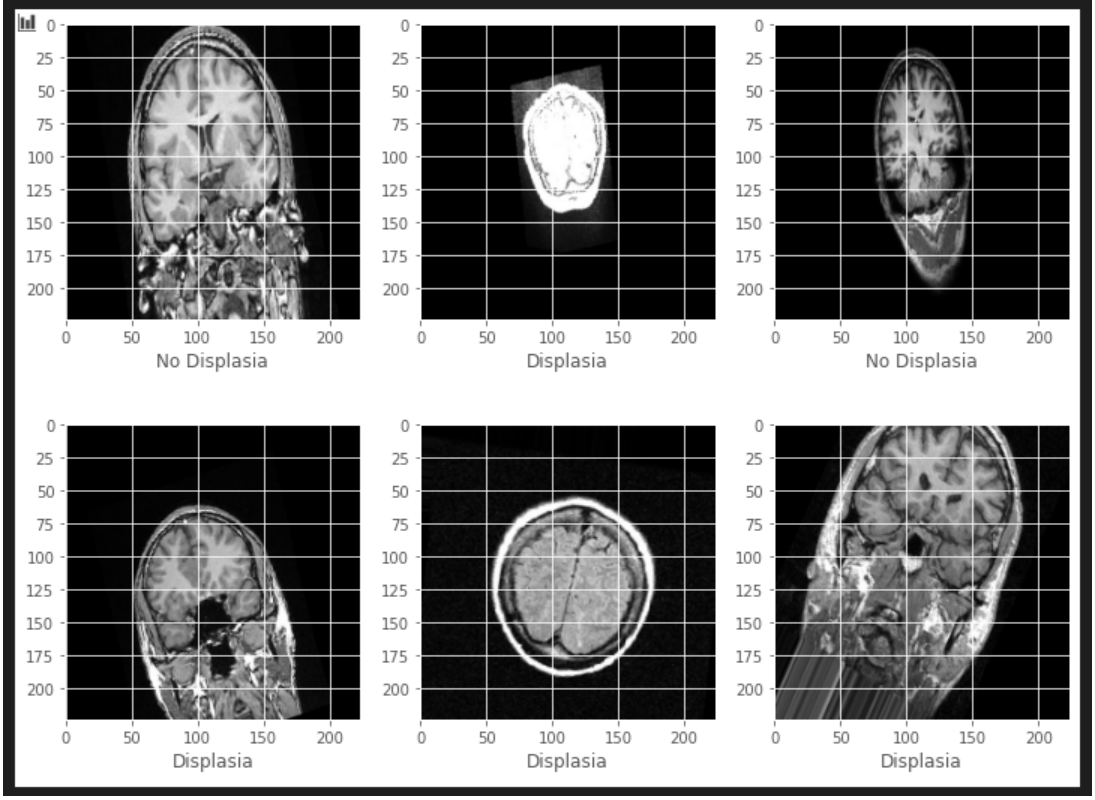

Figura 41: Imágenes de test generados.

```
effnet = EfficientNetB0(weights = "imagenet",include_top=False,input_shape=(img_size,img_size,3))

model = effnet.output
model = GlobalAveragePooling2D()(model)
model = Dropout(0.5)(model)
model = Dense(2,activation = "softmax")(model)

model = Model(inputs = effnet.input,outputs = model)

model.summary()
```

Figura 42: Manipulación de la red EfficientNet B0 y creación del modelo.

parámetro) no hayan mejorado los valores de métrica empleados. En estos tipos de ejercicios se suele introducir un valor conocido como early stop, que permitiría parar el entrenamiento cuando no se ha mejorado el entrenamiento en un número determinado de épocas, aquí no se ha introducido porque como nuestro número de imágenes era pequeño se producían muchos saltos de valores de precisión en el proceso de entrenamiento y cabía la posibilidad de que el modelo mejorara en cualquier momento.

Una vez realizado todo este proceso, ya estamos preparados para realizar el entrenamiento.

La información recogida en estos dos últimos párrafos podemos encontrarlo en forma de imagen en la Figura 43.

Por último se ha creado un código que nos muestra dos gráficos, el primero con los valores de precisión y el segundo con los valores de pérdida, que se han ido obteniendo a lo largo del proceso de entrenamiento, con la intención de estudiar la manera de hacer mejorar nuestro modelo. Se puede observar en la Figura 44.

Como se ha indicado previamente, todo este código es cómo se ha realizado la obtención del modelo en este proyecto, pero existen muchas otras formas de generar un modelo cambiando los parámetros introducidos e introduciendo más imágenes. Esta es la principal función de este apartado, saber dónde se puede modificar para continuar en una futura iteración y llegar a mejorar nuestro modelo.

```
model.compile(optimizer=Adam(lr=0.0001),loss="categorical_crossentropy",metrics = ["accuracy"])
checkpoint = ModelCheckpoint("effnetDEFINITIVO.h5",monitor="val_accuracy",save_best_only=True,mode="auto",verbose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3,
                              patience = 2, min_delta = 0.001,
                              mode = 'auto', verbose = 1)
```

```
   history = model.fit(X_train,y_train,epochs=15,validation_data = train_gen ,verbose=1,
       callbacks=[checkpoint,reduce_lr])
```

```
Epoch 1/15
24/24 [==============================] - 195s 8s/step - loss: 0.7229 - accuracy: 0.5612 - val_loss: 0.6618 - val_accuracy: 0.6160

Epoch 00001: val_accuracy improved from -inf to 0.61601, saving model to effnetDEFINITIVO.h5
Epoch 2/15
24/24 [==============================] - 184s 8s/step - loss: 0.5287 - accuracy: 0.7469 - val_loss: 0.6223 - val_accuracy: 0.6893

Epoch 00002: val_accuracy improved from 0.61601 to 0.68928, saving model to effnetDEFINITIVO.h5
Epoch 3/15
24/24 [==============================] - 183s 8s/step - loss: 0.4054 - accuracy: 0.8392 - val_loss: 0.5902 - val_accuracy: 0.7083

Epoch 00003: val_accuracy improved from 0.68928 to 0.70828, saving model to effnetDEFINITIVO.h5
Epoch 4/15
24/24 [==============================] - 183s 8s/step - loss: 0.3594 - accuracy: 0.8570 - val_loss: 0.5700 - val_accuracy: 0.6906

Epoch 00004: val_accuracy did not improve from 0.70828
Epoch 5/15
24/24 [==============================] - 183s 8s/step - loss: 0.2969 - accuracy: 0.8841 - val_loss: 0.5756 - val_accuracy: 0.6947

Epoch 00005: val_accuracy did not improve from 0.70828
```

```
   val_loss,val_acc = model.evaluate(train_gen)
   print(f"Validation Loss: {val_loss}")
   print(f"Validation Accuracy: {val_acc}")
```

```
24/24 [==============================] - 30s 1s/step - loss: 0.5508 - accuracy: 0.7232
Validation Loss: 0.5508129596710205
Validation Accuracy: 0.7232021689414978
```

Figura 43: Compilación del modelo y entrenamiento.

```
plt.style.use("ggplot")
plt.figure(figsize=(12,6))
epochs = range(1,16)
plt.subplot(1,2,1)
plt.plot(epochs,history.history["accuracy"],'go-')
plt.plot(epochs,history.history["val_accuracy"],'ro-')
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend(['Train','Val'],loc = "upper left")
plt.subplot(1,2,2)
plt.plot(epochs,history.history["loss"],'go-')
plt.plot(epochs,history.history["val_loss"],'ro-')
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(['Train','Val'],loc = "upper left")
plt.show()
```

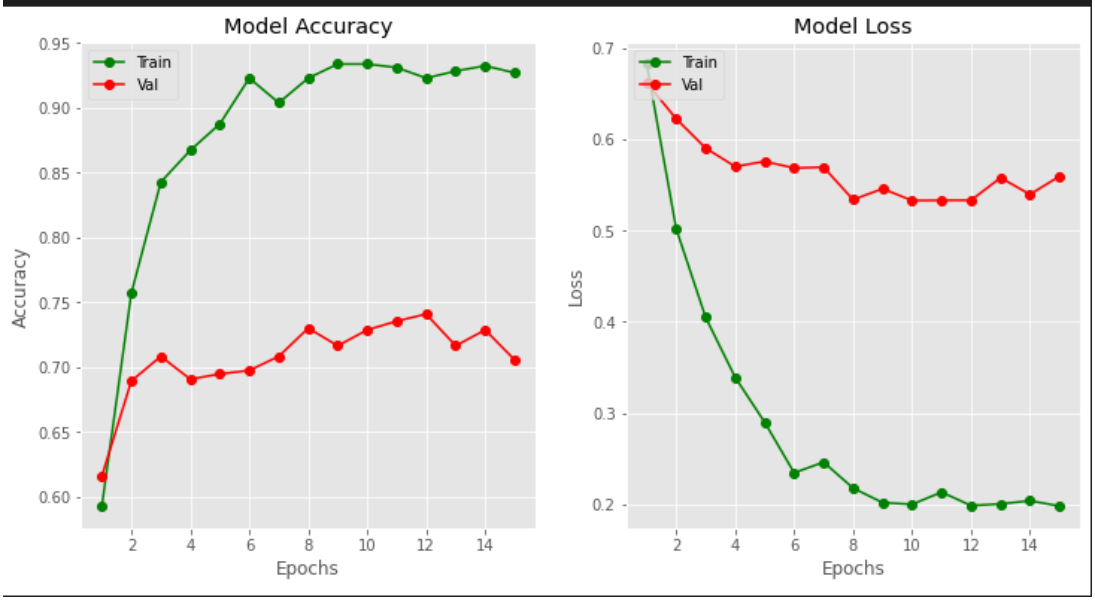

Figura 44: Gráfica de los valores obtenidos en el entrenamiento.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA